FernUniversität in Hagen
Fakultät für Mathematik und Informatik
Lehrgebiet Diskrete Mathematik und Optimierung

Diplomarbeit

b-Matching Spiele

von
Markus Werner

betreut von
Prof. Dr. Winfried Hochstättler

Großheirath im Mai 2009

Vorwort

An dieser Stelle möchte ich mich bei allen bedanken, die mich während der Erstellung dieser Arbeit unterstützt haben.

Mein besonderer Dank gilt hierbei vor allem Herrn Prof. Dr. Winfried Hochstättler, der mir während der gesamten Zeit geduldig mit Rat und Tat zur Seite stand.

Außerdem möchte ich mich bei meiner Familie und meiner Lebensgefährtin Kristina Höfer bedanken, die es mir durch ihr Verständnis und ihre Rücksichtnahme erst ermöglicht haben, mein Studium durchzuführen und diese Arbeit zu schreiben.

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

Auf dem Arbeitsmarkt seien n Firmen und m Arbeiter. Jede Firma besitzt einen individuellen Bedarf an Arbeitszeiteinheiten, für den Arbeiter gesucht werden. Jeder Arbeiter ist bereit, eine individuell festgelegte Anzahl an Arbeitszeiteinheiten zu arbeiten. Alle Marktteilnehmer besitzen eine Präferenzliste, nach der sie ihre potentiellen Vertragspartner beurteilen. So könnten zum Beispiel Arbeiter die Firmen nach dem ihnen angebotenen Gehalt beurteilen, während für die Firmen die Qualifikation und Leistungsfähigkeit der Bewerber das Beurteilungskriterium darstellt. Außerdem ist für jeden möglichen Arbeitsvertrag zwischen einer Firma i und einem Arbeiter j die maximale Anzahl an Arbeitszeiteinheiten festgelegt.

Gesucht ist nun eine Menge von Arbeitsverträgen, die alle hier genannten Beschränkungen berücksichtigt und mit der eine über alle Marktteilnehmer gesehen maximale Produktivität entsteht.

Mit diesem anschaulichen Beispiel lässt sich das Problem annähernd umschreiben, dessen algorithmische Lösung einen Großteil dieser Arbeit ausmachen wird. Es handelt sich um ein Matching-Problem zwischen disjunkten Mengen, das viele klassische Matching-Probleme verallgemeinert. Bevor wir dieses Problem mathematisch beschreiben können, sind allerdings noch einige Vorkenntnisse und Begriffserläuterungen notwendig. Beginnen möchte ich allerdings mit einem Überblick über die Entwicklungen in der Matching-Theorie, auf denen diese Arbeit aufbaut.

Bei Matching-Problemen versucht man allgemein gesprochen sogenannte *Agenten* unter Berücksichtigung spezieller Nebenbedingungen miteinander zu verknüpfen, was auch als *matchen* bezeichnet wird. Bei *zweiseitigen* Matchings gehört jeder Agent einer von zwei disjunkten Mengen an und ein Matching ist nur mit einem Agenten aus der anderen Menge möglich. Als Beispiel für solche zweiseitigen Matchings können Arbeitsverträge auf dem Arbeitsmarkt herangezogen werden - von Sonderfällen, in denen zum Beispiel Personen als Arbeiter und Arbeitgeber auftreten, einmal abgesehen. Hier gehört jeder Teilnehmer - man spricht in dem Zusammenhang in der Regel vereinfachend von Firmen und Arbeitern - entweder der Arbeitgeber- oder der Arbeitnehmerseite an. Sowohl die Firmen als auch die Arbeiter besitzen nun individuelle Präferenzen, mit welchem der Agenten sie einen Arbeitsvertrag schließen möchten. Als Beispiel für einen Markt, der nicht zweiseitig ist, kann der allgemeine Warenmarkt angese-

hen werden. Personen können hier sowohl als Käufer und gleichzeitig oder später auch als Verkäufer auftreten.

Zwei grundlegende Modelle wurden für die zweiseitigen Märkte eingeführt. Das Hochzeitsproblem von Gale und Shapley [10] und das sogenannte Zuweisungsspiel von Shapley und Shubik [26]. Beide Modelle enthalten unterschiedliche Bedingungen, die bei der Findung eines stabilen Matchings berücksichtigt werden müssen. Im Hochzeitsproblem besitzt jeder Agent einer Präferenzliste. Im Zuweisungsspiel ist jeder möglichen Kante ein Gewicht zugeordnet. Man kann dieses Gewicht als Geld interpretieren, welches als kontinuierliche Variable zwischen je zwei Partnern aufgeteilt werden muss. Das Zuweisungsspiel kann als das Problem interpretiert werden, ein maximal gewichtetes Matching in einem bipartiten Graphen zu finden. Die dazu notwendigen mathematischen Begriffe werden in Kapitel 2 eingeführt. Die (stabile) Lösung des Zuweisungsspiels heißt *Outcome* und beinhaltet neben einem Matching auch eine Zuteilung der Kantengewichte zu den jeweiligen Partnern.

Mit diesen beiden Modellen kann man recht gut Eigenheiten eines idealisierten Arbeitsmarktes simulieren. Das Modell von Gale und Shapley stellt die Situation dar, dass auf einem Arbeitsmarkt von Firmen Verträge mit festem Gehalt nach Tarifvertrag angeboten werden. Die Präferenzen der Arbeiter können sich dann aus dem zu erwartenden Einkommen ergeben. Arbeitsverträge mit flexiblem Gehalt können über das Zuweisungsspiel modelliert werden. Gehälter können hier zwischen Firmen und Arbeitern ausgehandelt werden, lediglich die Gesamtproduktivität aus einer möglichen vertraglichen Bindung von Firma und Arbeiter steht von Beginn an fest.

Eriksson und Karlander führen in [5] ein Modell ein, welches die beiden hier vorgestellten Modelle beinhaltet. Sie geben auch einen Algorithmus für die Ermittlung eines stabilen Outcomes an.

In der Folgezeit wurden immer wieder Arbeiten zu diesem Thema veröffentlicht. Neue Problemstellungen wurden dargelegt und neue, effizientere Algorithmen konnten entwickelt werden. In [13] stellen Hochstättler, Nickel und Schiess einen neuen, effizienten Algorithmus für eine Generalisierung des Modells von Eriksson und Karlander vor. Dieser Algorithmus ist einer der entscheidenden Ausgangspunkte dieser Arbeit. In Anlehnung an die Autorennamen werde ich diesen Algorithmus im Folgenden als *(klassischen) HNS-Algorithmus* bezeichnen.

Alle bislang hier angesprochenen Arbeiten weisen die Eigenheit auf, dass ein maximales Matching gesucht wird. Das heißt insbesondere, dass jeder Agent höchstens einen Partner besitzt.[1] Anstelle von Matchings sollen nun sogenannte b-Matchings untersucht werden. Hierbei wird jeder Agent x mit einer „Vielfachheit“ b_x versehen - daher auch die Namensgebung. Das bedeutet, dass Agenten auch mit mehreren Partnern gematcht sein können. Gleichzeitig ordnet man jeder Kante zwischen zwei Agenten x und y eine Kapazität k_{xy} zu.[2] Es ist klar, dass dadurch die Komplexität eines Problems in der Regel zunimmt. Die klassischen Matching-Probleme sind Sonderfälle solcher b-Matching-Probleme.

Mit dieser Arbeit sollen publizierte Arbeiten aus dem Themenbereich der b-

[1] In [10] stellen Gale und Shapley mit dem *College-Zuweisungsspiel* ein sogenanntes *many-to-one*-Matching vor. Hier dürfen Agenten einer der beiden disjunkten Mengen sogar mehrere Partnerschaften eingehen. Siehe dazu auch die Vorbemerkung in Kapitel 4.

[2] Nicht in allen b-Matching-Modellen werden Vielfachheiten auf Kanten zugelassen. In diesen Modellen kann man für alle Kanten $k_{xy} = 1$ annehmen.

Matchings vorgestellt und in Zusammenhang gebracht werden. Dabei habe ich mich auf den Bereich der bipartiten b-Matchings beschränkt. In diesem Bereich gibt es einige interessante und schnell nachvollziehbare Ansätze, während im allgemeinen Fall die Modelle sehr schnell sehr komplex werden. Außerdem wird als Generalisierung des klassischen HNS-Algorithmus ein *b-HNS-Algorithmus* entwickelt. Nach einigen Begriffsbestimmungen im folgenden Kapitel werden im Kapitel 3 die klassischen Matching-Probleme und deren Algorithmen kurz vorgestellt. Das Kapitel 4 stellt einen Überblick über Arbeiten zu b-Matchings dar. Das Kapitel 5 schließlich beinhaltet die Entwicklung, den Beweis und Anwendungsmöglichkeiten des b-HNS-Algorithmus.

Kapitel 2

Definitionen und Notationen

In diesem Kapitel sollen einige grundlegende Begriffe der Graphentheorie vorgestellt werden. Die Definitionen wurden dabei [1], [19], [25] und [30] entnommen. Dabei wurden einzelne Formulierungen so geändert, dass der Zusammenhang zu den anderen Definitionen ersichtlich wird. Leser, die bereits ein gewisses Grundwissen in diesem Bereich besitzen, sollten vor allem Augenmerk auf die hier definierten Schreibweisen legen. Hinweise zu den Begriffsbestimmungen werden in diesem Kapitel *kursiv* dargestellt.

2.1 Graphentheoretische Grundbegriffe

Ein **(ungerichteter) Graph** $G = (V, E)$ besteht aus einer **Knotenmenge** V und einer **Kantenmenge** E, wobei jeder **Kante** $e \in E$ von G zwei (nicht notwendig verschiedene) **Knoten** aus V zugeordnet sind. Im weiteren Verlauf des Textes wird eine Kante e häufig in der Form (i, j) geschrieben, wobei $i, j \in V$ genau die beiden **Endknoten** der Kante e sind.

Wenn $i \in V$ ein Endknoten einer Kante e ist, so heißen i und e **inzident**. Zwei Knoten $i, j \in V$ heißen **adjazent** oder benachbart, wenn im Graphen G eine Kante $e = (i, j)$ existiert. Zwei Kanten heißen **adjazent**, wenn ein Knoten $v' \in V$ existiert, der Endknoten beider Kanten ist.

Ein **gerichteter Graph** $G = (V, E)$ besteht aus einer Knotenmenge V und einer Pfeilmenge E, sodass jedem **Pfeil** $e = (u, v)$ eindeutig ein geordnetes Paar (u, v) von Knoten aus V zugeordnet ist. Der Knoten u heißt **Anfangsknoten**, v der **Endknoten** des Pfeils $e = (u, v)$. Gerichtete Graphen werden auch als **Digraphen** bezeichnet.

In der Notation für Kanten und Pfeile gibt es in der obigen Definition keine Unterschiede. In dieser Arbeit werden sowohl Kanten als auch Pfeile verwendet. Es wird immer darauf hingewiesen, wenn es sich bei der Schreibweise (i, j) *nicht um eine Kante, sondern um einen Pfeil handelt.*

Ein **bipartiter Graph** $G = (V, E)$ besitzt eine Knotenmenge V, die in zwei disjunkte Teilmengen U und W zerfällt, und jede Kante $e \in E$ besitzt genau einen Endknoten in U und einen Endknoten in W. Man verwendet für die Kno-

tenmenge die Schreibweise $V = U \dot{\cup} W$. Ein bipartiter Graph $G = (U \dot{\cup} W, E)$ heißt **vollständig**, wenn jeder Knoten aus U adjazent ist zu jedem Knoten aus W.

Gerade die vollständigen bipartiten Graphen sind es, die im Laufe der Arbeit von Interesse. Solche Graphen bilden das Ausgangsproblem ab, Beziehungen zwischen zwei disjunkten Mengen, wie zum Beispiel Firmen und Arbeitern, unter bestimmten Nebenbedingungen herzustellen. In dieser Arbeit werden die Knoten eines solchen Graphen auch als Agenten bezeichnet. Auch wird häufiger der Begriff Firma beziehungsweise Arbeiter für die Knoten der beiden disjunkten Knotenmengen verwendet.

Eine Kantenteilmenge $F \subseteq E$ eines Graphen $G = (V, E)$ heißt ein **Matching** von G, wenn keine zwei Kanten aus F einen gemeinsamen Endknoten besitzen. Ein **maximales Matching** ist ein Matching mit maximaler Kantenzahl. Ein Matching, in dem alle Knoten eines Graphen Endknoten (genau) einer Kante sind, heißt **perfekt**.

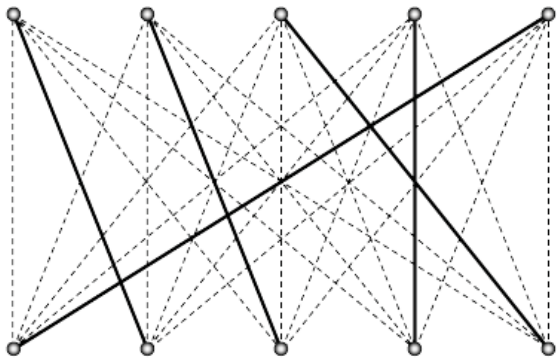

Abbildung 2.1: Ein perfektes Matching in einem bipartiten Graphen.

Die obige Abbildung illustriert ein Matching in einem bipartiten Graphen. Gestrichelte Linien entsprechen dabei möglichen Partnerschaften zwischen zwei Agenten. Durchgezogene Linien stellen tatsächliche Partnerschaften dar. Jeder Agent besitzt genau eine Partnerschaft.

Für die Überführung eines Matching-Problems in ein lineares Optimierungsproblem definieren wir noch den **charakteristischen Vektor** einer Teilmenge $F \subseteq E$: $\chi^F = \left(\chi_e^F\right)_{e \in E} \in \mathbb{R}^E$ heißt charakteristischer Vektor von $F \subseteq E$, falls für alle $e \in E$ gilt: $\chi_e^F = 1$, falls $e \in F$, und $\chi_e^F = 0$ sonst.

2.2 Gewichtete bipartite Graphen, b-Matchings

Wie in der Einleitung bereits beschrieben, gibt es bei den Matching-Spielen und b-Matching-Spielen Nebenbedingungen, die erfüllt sein müssen, damit ein Matching zulässig oder maximal im Sinne des zu Grunde liegenden Modells ist.

Hierbei handelt es sich zum Beispiel um Präferenzen, die jede Firma und jeder Arbeiter gegenüber einer Kante zu einem möglichen Partner hat. Diese Präferenzen schlagen sich in einer sogenannten Gewichtung der einzelnen Kanten nieder.

Ein Graph $G = (V, E, \alpha)$ heißt **gewichtet**, wenn er eine Gewichtsfunktion $\alpha : E \to \mathbb{R}_+$ besitzt. Sei M ein Matching im Graphen $G = (V, E, \alpha)$. Das **Gewicht eines Matchings** M wird definiert als $\alpha(M) := \sum_{e \in E} \alpha(e)$. M heißt ein **maximal gewichtetes Matching**, falls $\alpha(M) \geq \alpha(M')$ für alle anderen Matchings M' von G gilt. Maximal gewichtete Matchings bezeichnet man auch als **optimal**.

Sei nun $G = (V, E)$ ein Graph. Man kann nun jedem Knoten $v \in V$ eine Zahl $b_v \in \mathbb{N}_+$ zuordnen. b_V heißt **Kapazität des Knotens** v. Entsprechend kann jeder Kante $e \in E$ eine Zahl $k_e \in \mathbb{N}_+$ zuordnen. k_e wird als **Kantenkapazität** von e bezeichnet. Für eine beliebige Teilmenge $A \subseteq V$ sei

$$\delta(A) := \{\{v, u\} \in E | v \in A, u \in V \backslash A\}.$$

Ein b-**Matching** ist damit eine Funktion $f : E \to \mathbb{Z}_+$ mit

$$f(e) \leq k_e \ \forall e \in E$$
$$\sum_{e \in \delta(v)} f(e) \leq b_v \ \forall v \in V$$

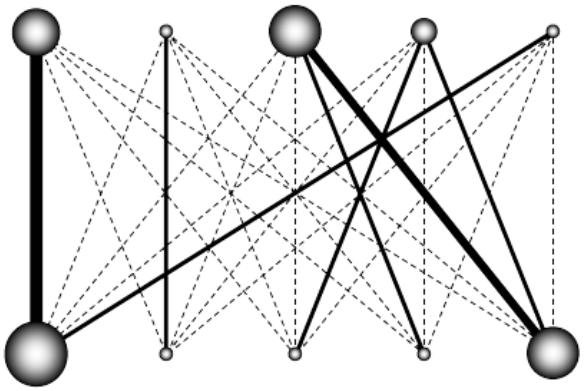

Abbildung 2.2: Ein maximales b-Matching in einem bipartiten Graphen.

Auch wenn hier ein Matching über eine Teilmenge von Kanten definiert wurde, ein b-Matching aber als eine Funktion, so ist dennoch der große Zusammenhang zwischen den beiden Begriffen ersichtlich. Beim b-Matching sind lediglich auch „Vielfachheiten" auf den Kanten möglich, außerdem können Knoten gleich mehrere inzidente Kanten besitzen. Ist $b_v = 1$ für alle Knoten aus V, dann stimmen die Begriffe Matching und b-Matching überein.

Ein b-Matching heißt **perfekt** falls $\sum_{e \in \delta(v)} f(e) = b_v \ \forall v \in V$ gilt.

Im Unterschied zu Abbildung 2.1 wird in Abbildung 2.2 ein b-Matching dargestellt. Ein größerer Knoten hat dabei eine größere Vielfachheit als ein kleinerer Knoten. Dickere durchgezogene Linien entsprechen Partnerschaften mit höherer Vielfachheit.

2.3 Pfade und Netzwerke

Eine **Kantenfolge** in einem Graphen $G = (V, E)$ ist eine Folge

$$v_1, e_1, v_2, e_2, v_3, \ldots, v_{k-1}, e_{k-1}, v_k$$

von Knoten aus V und Kanten aus E, sodass jede Kante e_i für $i = 1, \ldots, k-1$ jeweils die Endknoten v_i und v_{i+1} besitzt. Man spricht auch davon, dass diese Kantenfolge v_1 mit v_k **verbindet**. Die **Länge** der Kantenfolge ist die Anzahl der Kanten dieser Folge. Eine Kantenfolge heißt **Weg** oder auch **Pfad**, wenn jeder Knoten aus V höchstens einmal in dieser Folge auftritt.

Sei $G = (V, E)$ ein Digraph. Ein Knoten $s \in V$ heißt **Quelle** in G, wenn s von keinem Pfeil aus E Endknoten ist. Ein Knoten $t \in V$ heißt Senke, wenn t von keinem Pfeil aus E Anfangsknoten ist.

Ein **(Fluss-)Netzwerk** $N = (G, c, s, t)$ ist ein gerichteter Graph $G = (V, E)$ mit einer Quelle s und einer Senke t und einer Kapazitätsfunktion $c : V \times V \to \mathbb{R}$, so dass gilt

1. $s \neq t$
2. $c(v, w) \geq 0$ für alle $v, w \in V$
3. $c(v, w) > 0 \Leftrightarrow (v, w) \in E$

Für die Suche möglichst großer Matchings werden recht häufig Alternierungen und Augmentierungen benötigt. Sei M ein Matching in einem Graphen $G = (V, E)$. Ein Pfad in G, der in einem ungematchten Knoten aus V beginnt und dann abwechselnd Kanten aus $E \backslash M$ und aus M enthält, heißt **alternierender Pfad** (bezüglich M). Ein alternierender Pfad, der in einem ungematchten Knoten aus V endet, wird als **augmentierenden Pfad** bezeichnet. Sei P ein alternierender oder augmentierender Pfad in G mit Matching M. Unter **Alternierung** beziehungsweise **Augmentierung** wird der Vorgang verstanden, dass alle Kanten aus P, die bislang zu M gehörten, aus M gestrichen werden, während alle anderen Kanten aus P zu M hinzugefügt werden. Man beachte, dass bei Alternierungen die Mächtigkeit von M gleich bleibt, während bei Augmentierungen die Mächtigkeit von M um 1 zunimmt.

2.4 Algorithmen und Komplexität

Nachdem ein recht großer Teil dieser Arbeit auf die Konstruktion und Erläuterung von Algorithmen entfällt, werden noch Begriffe aus diesem Bereich eingeführt:

Unter einem **Algorithmus** (auch Lösungsverfahren) versteht man eine genau

definierte Handlungsvorschrift zur Lösung eines Problems oder einer bestimmten Art von Problemen in endlich vielen Schritten.

Bei der Konstruktion eines Algorithmus wird auch ein besonderes Augenmerk darauf gelegt, mit wie vielen Schritten der Algorithmus zum Ergebnis kommt. Dabei ist die genaue Anzahl der Schritte in der Regel gar nicht so wichtig, vielmehr interessiert die Größenordnung dieser Zahl in Abhängigkeit von der Eingabegröße. Zur Beschreibung solcher Größenordnungen ist die $\mathcal{O}$-**Notation** sehr nützlich.

Seien $f, g : \mathbb{N}_0^r \to \mathbb{R}$. Gibt es $B, C \in \mathbb{N}_0$, so dass für alle $(n_1, \ldots, n_r \in \mathbb{N}_0^r)$ mit $n_i \geq B$ für alle $1 \leq i \leq r$ gilt:

1. $f(n_1, \ldots, n_r) > 0$ und $g(n_1, \ldots, n_r) > 0$, und
2. $f(n_1, \ldots, n_r) < C \cdot g(n_1, \ldots, n_r)$

Dann heißt f **durch** g **beschränkt** und man schreibt $f = \mathcal{O}(g)$.

Die $\mathcal{O}$-Notation ist sehr hilfreich, um das Verhalten von Algorithmen für große Eingaben abzuschätzen. Dazu wird die Größenordnung der Schritte eines Algorithmus durch eine Funktion abgeschätzt. In dem Zusammenhang seien mit der obigen Notation $n_1, \ldots, n_r$ die Größenordnungen der r Eingabewerte eines Algorithmus (zum Beispiel die Anzahl der benötigten Bits bei computertechnischer Umsetzung). Existiert ein g gemäß der vorangehenden Definition, dann besitzt der Algorithmus die **Laufzeit** $\mathcal{O}(g)$.

Ein Algorithmus heißt **polynomiell** oder auch **effizient**, wenn g ein Polynom ist. Ein Algorithmus heißt **pseudopolynomiell**, wenn die Laufzeit des Algorithmus in polynomieller Abhängigkeit zum numerischen Wert der Eingabe steht.

Nun wird man immer bestrebt sein, effiziente Algorithmen zu entwickeln. Denn solche Algorithmen stellen sicher, dass auch große Eingabemengen nicht dazu führen, dass die Laufzeit des Algorithmus „über alle Maßen“ wächst. Insbesondere sollte ein exponentielles Wachstum stets vermieden werden. Außerdem lässt die $\mathcal{O}$-Notation eine recht gute Vergleichbarkeit zwischen zwei Algorithmen zu einer Problemstellung zu.

Kapitel 3

Klassische Matching-Spiele

Wie bereits in der Einleitung erwähnt, stellen b-Matching-Spiele in der Regel Verallgemeinerungen klassischer Matching-Spiele dar. Aus dem Grund sollen einige klassische Matching-Probleme vorgestellt werden. Es werden jeweils nur die Ideen für die Modellierung und die Lösungsstrategien dargestellt. Für nähere Informationen sei auf die jeweiligen Arbeiten verwiesen.

3.1 Das Hochzeitsproblem von Gale und Shapley

Beim Hochzeitsproblem [10] sind zwei disjunkte Mengen $M := (m_1, \ldots, m_n)$ und $W := (w_1, \ldots, w_n)$ gegeben. Gale und Shapley gehen von der gleichen Mächtigkeit von M und W aus. So ist sichergestellt, dass alle Elemente gematcht werden können. Das Modell kann aber einfach auf unterschiedlich große Mengen M und W übertragen werden. Dann werden Dummyknoten notwendig. Solche Knoten werden von möglichen Partnern am wenigsten präferiert. Knoten, die am Ende mit einem Dummyknoten gematcht sind, sind im Ausgangsproblem ungematcht. Anstelle von Firmen und Arbeitern verwenden Gale und Shapley als Beispiel Männer und Frauen im heiratsfähigen Alter, woraus sich auch der Name dieses Modells ableitet.

Jedes Element von M und jedes Element von W besitzt eine Präferenzliste der Elemente von W beziehungsweise M. Eine solche Präferenzliste lässt sich leicht in Matrizen mit Einträgen aus $\mathbb{R}_+$ übertragen. Seien dafür $A := (a_{ij}), B := (b_{ij}) \in \mathbb{R}_+^{n \times n}$ als die Präferenzmatrizen von M beziehungsweise W definiert. Seien $i \in M$ und $j_1, j_2 \in W$ beliebig. Falls i j_1 gegenüber j_2 bevorzugt, dann gilt $a_{ij_1} > a_{ij_2}$. Falls i j_1 und j_2 als gleich ansieht, dann gilt $a_{ij_1} = a_{ij_2}$. Falls i ebenso gerne unverheiratet bleibt als mit i_1 verheiratet zu werden, ist $a_{ij_1} = 0$. Gleiches gilt für die Einträge in B. Falls dort also zum Beispiel $b_{i_1 j} > b_{i_2 j}$ ist, dann bevorzugt $j \in W$ i_1 vor i_2. Diese Präferenzmatrizen sind es, die im Laufe der Arbeit immer wieder von Relevanz sein werden.

Beim Hochzeitsproblem geht es nun darum, ein stabiles Matching der Elemente aus M und W zu finden. Dabei gehen Gale und Shapley davon aus, dass M und W gleich viele Elemente besitzen, alle Präferenzen strikt positiv sind und jeder Agent keine gleichen Präferenzen gegenüber zwei möglichen Partnern hat. Es ist jedoch recht einfach auf diese Einschränkungen zu verzichten. Bei gleich gewichteten Partnern wird willkürlich einer ausgewählt, ein Agent besitzt

nur dann die Bereitschaft eine bestehende Partnerschaft aufzugeben, wenn die Präferenz für einen potentiellen neuen Partner echt größer ist als für den aktuellen. Falls auch negative Präferenzen existieren, legt man als zusätzliche Regel fest, dass nur Angebote an Partner unterbreitet und akzeptiert werden, wenn die Präferenz strikt positiv ist. Im Ergebnis können damit auch „Singles“ entstehen. Ein Matching heißt dabei stabil, wenn für je zwei Elemente $i \in M$ und $j \in W$ gilt: Falls i und j im vorliegenden Matching nicht miteinander gematcht sind, dann wären nicht beide Agenten lieber miteinander gematcht als mit ihrem derzeitigen Partner.

Gale und Shapley beweisen algorithmisch, dass ein solches stabiles Matching immer existiert. Dieser Algorithmus ist unter dem Namen *Men-propose-Women-dispose-Algorithmus* bekannt. Jeder Mann aus M macht seiner Lieblingsfrau aus W ein Angebot. Alle Frauen, die mehr als ein Angebot bekommen, weisen alle Angebote zurück bis auf dasjenige, das sie am meisten bevorzugen. Diejenigen Männer, die nach diesem Schritt keinen Partner besitzen, streichen die Frau, die ihr Angebot abgewiesen hat, von ihrer Liste und unterbreiten der Frau, die sie nun am meisten bevorzugen, ein Angebot. Dieser Zyklus aus Angebotsunterbreitung durch die Männer und Angebotsprüfung und Ablehnung durch die Frauen wiederholt sich so lange, bis ein stabiles Matching erreicht ist. Falls $|M| = |W|$ und alle Präferenzen strikt positiv sind, besitzt zum Ende des Algorithmus jeder Agent einen Partner.

Dass man mit diesem Algorithmus tatsächlich ein stabiles Matching erhält, lässt sich leicht zeigen. Seien dazu $i \in M$ und $j \in W$ nach Durchführung des Algorithmus nicht gematcht. Angenommen, i bevorzugt dennoch j vor seinem derzeitigen Partner. Dann muss i im Verlaufe des Algorithmus ein Angebot an j unterbreitet haben und dieses wurde von j abgelehnt. Es konnte aber nur deswegen abgelehnt werden, weil j ein besseres Angebot vorlag. Und damit sind wir bereits fertig.

3.2 Das Zuweisungsspiel von Shapley und Shubik

Das Zuweisungsspiel [26] ist ein kooperatives Spiel, bei dem die eigentlich diskrete Variable Geld als kontinuierliche Variable modelliert wird. Die Entscheidungsfindung eines einzelnen Agenten orientiert sich ausschließlich an monetären Gesichtspunkten. Wie wir sehen werden, läuft dieses Problem auf die Ermittlung eines maximalen Matchings in einem bipartiten Graphen hinaus. Neben dem eigentlichen Problem ist insbesondere die von Kuhn vorgestellte Ungarische Methode zur Ermittlungen einer stabilen Lösung von hoher Relevanz.

Für die Modellierung des Zuweisungsspiels seien $P = \{1, \ldots, n\}$ und $Q = \{1, \ldots, m\}$ zwei endliche disjunkte Mengen. Diese Mengen kann man zum Beispiel als Verkäufer und Käufer auf dem Warenmarkt oder als Firmen und Arbeiter auf dem Arbeitsmarkt interpretieren. Für jede mögliche Koalition $\{i, j\}$ mit $i \in P$ und $j \in Q$ sei α_{ij} der Wert dieser Koalition. Kommt eine solche Koalition zu Stande, i und j sind also im zugehörigen Graphen gematcht, dann kann die Produktivität α_{ij} frei zwischen i und j aufgeteilt werden. Man spricht in dem Zusammenhang von *side payments*, nachdem bildlich gesprochen eine Firma einem Arbeiter, mit dem eine Koalition besteht, einen Teil seines Profits abgeben kann, damit diese Koalition bestehen bleibt. Den Anteil, den ein Knoten $i \in P$ beziehungsweise $j \in Q$ erhält, wird mit u_i beziehungsweise v_j

bezeichnet. Wir schreiben $u := (u_i)_{i \in P}$ und $v := (v_j)_{j \in Q}$ und nennen (u, v) ein *Payoff* des Problems.

Dieses Problem ist vergleichbar mit dem Problem, in einem bipartiten Graphen mit Knotenmenge $P \dot{\cup} Q$ und der Kantengewichtsmatrix $A = (\alpha_{ij}) \in \mathbb{R}^{n \times m}$ ein maximal gewichtetes Matching zu finden. Wir können dieses Problem in ein lineares Programm überführen. Sei dazu bei einem gegebenen Matching M $x = (x_{ij}) \in \mathbb{R}^{n \times m}$ die Matrix, deren Einträge x_{ij} genau dann 1 sind, wenn i und j in M gematcht sind. Andernfalls seien die Einträge 0. Das lineare Programm (LP) lautet damit:

$$\begin{aligned} \max \quad & \sum_{i \in P} \sum_{j \in Q} \alpha_{ij} \cdot x_{ij} \\ \text{unter den Nebenbedingungen} \quad & (a) \ \sum_{i \in P} x_{ij} \leq 1 \\ & (b) \ \sum_{j \in Q} x_{ij} \leq 1 \\ & (c) \ x_{ij} \geq 0 \end{aligned}$$

Dieses lineare Programm kann nun dualisiert werden und man erhält das duale Programm (DLP):

$$\begin{aligned} \min \quad & \sum_{i \in P} u_i + \sum_{j \in Q} v_j \\ \text{unter den Nebenbedingungen} \quad & (a) \ u_i \geq 0 \\ & (b) \ v_j \geq 0 \\ & (c) \ u_i + v_j \geq \alpha_{ij} \end{aligned}$$

Man kann zeigen, dass (LP) eine ganzzahlige Lösung besitzt [4], woraus folgt, dass auch (DLP) eine optimale Lösung besitzt und die Optima von (LP) und (DLP) sind gleich. Falls also x die zum optimalen gewichteten Matching gehörende Matrix ist und (u, v) eine Lösung des (DLP) ist, dann gilt

$$\sum_{i \in P} u_i + \sum_{j \in Q} v_j = \sum_{i \in P} \sum_{j \in Q} \alpha_{ij} \cdot x_{ij} = \nu(P \dot{\cup} Q) \tag{3.1}$$

Ein *Outcome* besteht aus einem Matching M und einem Payoff (u, v). Ein Payoff und das zugehörige Outcome heißen zulässig, wenn die linke Gleichung aus 3.1 erfüllt ist. Ein zulässiges Outcome heißt stabil, wenn $u_i + v_j \geq \alpha_{ij}$ für alle $(i, j) \in P \times Q$ gilt.

Die Stabilitätsbedingung im Zuweisungsspiel ähnelt stark der Stabilitätsbedingung des Hochzeitsproblems. Angenommen, es liegt ein Matching vor, in dem $(i, j) \in P \times Q$ nicht gematcht sind und es gilt $u_i + v_j < \alpha_{ij}$. Dann ist es für i und j vorteilhaft, sich von ihrem derzeitigen Partner zu trennen, da durch die Koalition von i und j der Wert von u_i und v_j gesteigert werden kann.

Die Ungarische Methode

Als Lösungsmethode für das Zuweisungsspiel stellt Kuhn die sogenannte Ungarische Methode vor [18]. Dieser Algorithmus stellt eine wichtige Lösungsstrategie

für solche Probleme dar. Er findet in modifizierter Weise in vielen verwandten Algorithmen Anwendung, so auch im b-HNS-Algorithmus. Aus diesem Grund soll die Idee der Ungarischen Methode hier kurz skizziert werden.

Für die Ungarische Methode muss ein Matching M gegeben sein. Dabei kann auch $M = \emptyset$ gelten. Man kann von $|P| = |Q| = n$ ausgehen, andernfalls werden Dummyknoten eingefügt. Setze $u_i = \max_{j \in Q} \alpha_{ij}$ für alle $i \in P$ und $v_j = 0$ für alle $j \in Q$.

Aus dem bipartiten Graphen wird nun ein Digraph $H_{u,v}$ konstruiert. Betrachte alle Kanten (i, j), für die $\alpha_{ij} = u_i + v_j$ gilt. Orientiere diese Pfeile von P nach Q, falls i und j gematcht sind und von Q nach P sonst. Entsprechend kann man aus $H_{u,v}$ wieder ein Matching in G konstruieren. Mit [16] gilt die folgende Aussage:

Lemma

Wenn in $H_{u,v}$ ein perfektes Matching vorliegt, dann ist das sich aus $H_{u,v}$ ergebende Matching in G ein optimales Matching.

Im Schritt 1 wählt man einen in G ungematchten Knoten i' aus P und sucht einen alternierenden Pfad maximaler Länge in $H_{u,v}$. Diesen Pfad alterniert beziehungsweise augmentiert man dann, wodurch i' gematcht wird. Dieser Schritt wird wiederholt, bis kein solcher Pfad mehr existiert.

Falls man so kein perfektes Matching in $H_{u,v}$ erhält, führt man Schritt 2 aus. Dazu wählt man wieder einen ungematchten Knoten $i' \in P$ aus. Alle Knoten, die in $H_{u,v}$ von i' aus erreichbar sind, fügt man zu den Mengen $\overline{P} \subseteq P$ beziehungsweise $\overline{Q} \subset Q$ hinzu. Dann ermittelt man

$$\delta := \min \left\{ u_i + v_j - \alpha_{ij} | i \in \overline{P}, j \notin \overline{Q} \right\}$$

Bei allen Knoten $i \in \overline{P}$ subtrahiert man δ von u_i, während man bei allen Knoten $j \in \overline{Q}$ δ zu v_j addiert. Dadurch wird die Summe $\sum(u_i + v_j)$ reduziert, oder es kommt mindestens ein neuer Pfeil in $H_{u,v}$ hinzu. Anschließend beginnt man wieder mit Schritt 1. Der Algorithmus terminiert und besitzt eine Laufzeit von $\mathcal{O}\left(n^3\right)$.

3.3 Das Modell von Eriksson und Karlander

Eriksson und Karlander führen die beiden vorangegangenen Modelle zu einem einzigen Modell zusammen [5]. Außerdem geben sie im Beweis der Existenz eines stabilen Outcomes einen pseudopolynomiellen Algorithmus an. Sotomayor stellt einen anderen Beweis vor [28], der in [15] zu einem polynomiellen Algorithmus führt. Das Modell von Eriksson und Karlander soll nun kurz vorgestellt werden.

Ausgangspunkt ist ein bipartiter Graph mit Knotenmenge $P \dot{\cup} Q$. Es kann wieder von $|P| = |Q| = n$ ausgegangen werden, andernfalls werden entsprechende Dummyknoten hinzugefügt. Jedem Paar $(i, j) \in P \times Q$ wird ein Paar nichtnegativer reeller Zahlen (β_i, γ_j) zugeordnet. Die Idee ist, dass jede mögliche Partnerschaft (i, j) eine Produktivität von $\alpha_{ij} = \beta_{ij} + \gamma_{ij}$ besitzt. Die Aufteilung dieser Produktivität auf die beiden Agenten ist damit allerdings noch nicht festgelegt.

Für jeden Agenten steht von Beginn an fest, ob es sich um einen rigiden oder flexiblen Agenten handelt. Eine Kante zwischen zwei Agenten i und j ist rigide, wenn einer der beiden Agenten rigide ist, andernfalls ist die Kante flexibel. Sei also $\mathcal{R}$ die Menge der (möglichen) rigiden Kanten und $\mathcal{F}$ die Menge der (möglichen) flexiblen Kanten, dann gilt $\mathcal{R} \cup \mathcal{F} = P \times Q$. Bei einer rigiden Kante (i, j) erhält der Agent i aus P die Produktivität β_{ij} und der Agent j aus Q die Produktivität γ_{ij}. Bei einer flexiblen Kante wird die Produktivität α_{ij} der Kante frei unter den beteiligten Agenten aufgeteilt. Es werden also bei rigiden Kanten die Eigenschaften des Hochzeitsproblems und bei flexiblen Kanten die des Zuweisungsspiels angewendet.

Entsprechend werden die Zulässigkeits- und Stabilitätsbedingungen abhängig von der Art der (möglichen) Kanten zwischen zwei Agenten formuliert.

3.4 Der klassische HNS-Algorithmus

In ihrer Arbeit *Mixed Matching Markets* [13] stellen Hochstättler, Nickel und Schiess eine Verallgemeinerung des Modells von Eriksson und Karlander vor. In ihrem Modell steht nicht von Beginn an fest, ob eine Kante zwischen zwei Agenten rigide oder flexibel ist. Dies kann von jedem möglichen Paar von Agenten entschieden werden. Außerdem stellen sie einen Algorithmus vor, der in $\mathcal{O}(n^4)$ ein stabiles Outcome erzeugt.

Sei $G = (P \dot{\cup} Q, E)$ ein bipartiter Graph, wobei P und Q aus jeweils n Elementen bestehen. Zur Anschaulichkeit werden die Agenten aus P Firmen und die Agenten aus Q auch Arbeiter genannt. Die Matrizen $A = (a_{ij}), B = (b_{ij}), C = (c_{ij}) \in \mathbb{R}_+^{n \times n}$ stellen die Präferenzmatrizen dar. Seien $i \in P, j \in Q$ und die Kante (i, j) sei rigide. Dann gilt in einem zulässigen Outcome für die Profite $u_i = a_{ij}$ und $v_j = b_{ij}$. Falls die Kante (i, j) flexibel ist, gilt für die Profite die Gleichung $u_i + v_j = c_{ij}$.

Neben den bekannten Payoffs (u, v) und einem Matching M beinhaltet das Outcome dieses Algorithmus auch eine Flexibilitätsabbildung $f : \{1, \ldots, n\} \to \{0, 1\}$, für die gilt: $f(i) = 0$, falls die Kante mit Endknoten i rigide ist, und $f(i) = 1$, falls diese Kante flexibel ist.

Ein zulässiges Outcome heißt stabil, wenn zusätzlich für alle $1 \leq i \leq n$ und $1 \leq j \leq n$

$$u_i + v_j \geq c_{ij}$$

und

$$u_i \geq a_{ij} \text{ oder } v_j \geq b_{ij}$$

gilt.

Der klassische HNS-Algorithmus, der ein stabiles Outcome für dieses Modell erzeugt, verwendet die folgende Idee, die sich von den bekannten Algorithmen für solche Probleme abhebt:

Erzeuge im ersten Schritt ein Matching maximalen Gewichts, in dem nach Möglichkeit von jedem Knoten aus P eine Kante ausgeht. Die Knoten in Q können dabei durchaus auch mehrfach gematcht sein. Allerdings dürfen Knoten mit mehreren Kanten nur inzident zu flexiblen Kanten sein. Dabei erhöhen sich die Profite aus flexiblen Kanten für Arbeiter, falls ein entsprechend profitables rigides Angebot verliegt. Der Algorithmus orientiert sich hier in weiten Strecken am *Men-propose-Women-dispose-Algorithmus*. Nach Abschluss der ersten Phase

sind nur solche Arbeiter ungematcht, die in jedem stabilen Outcome ungematcht sind.

Im zweiten Schritt folgen Alternierungen beziehungsweise Augmentierungen in einem vergleichbar zum Zuweisungsspiel definierten Digraphen. Dabei ist allerdings im Unterschied zur oben vorgestellten Ungarischen Methode keine Firma, sondern ein Arbeiter mit mehr als einem vorliegenden Angebot Ausgangsknoten für zu alternierende Pfade.

Existieren keine solchen Pfade mehr, werden im dritten Schritt die Profite aus flexiblen Kanten analog zur Ungarischen Methode manipuliert. Abschließend werden ungematchte Firmen und Arbeiter rigide gematcht.

Wenn der Algorithmus terminiert, besitzt jeder Agent genau eine Kante und das Matching ist stabil.

Kapitel 4

b-Matching-Modelle

In diesem Kapitel sollen einige Arbeiten über b-Matching-Probleme und deren Lösungsansätze vorgestellt werden. Außerdem wird jeweils untersucht, wo die Unterschiede und die Gemeinsamkeiten zum im Kapitel 5 vorgestellten Modell liegen.

Bereits in ihrer Arbeit *College Admissions and the Stability of Marriage* [10] stellen Gale und Shapley ein Modell vor, bei dem zumindest für einen Teil der Knoten Vielfachheiten erlaubt sind. Es handelt sich um das sogenannte *Collegezuweisungsproblem*, bei dem Colleges gleich mehrere Studenten aufnehmen möchten, jeder Student aber höchstens einem College zugeordnet wird. In der Literatur (siehe unter anderem [23]) werden solche Probleme auch *many-to-one-matchings* genannt. Der Algorithmus, den Gale und Shapley hier anbieten, ist eng an den *Men-propose-Women-dispose-Algorithmus* angelehnt. Man kann die Studenten als *men* und die Colleges als *women* auffassen und im „dispose-Teil" des Algorithmus berücksichtigen, dass Colleges die am wenigsten bevorzugten Studenten ablehnen, sobald ihnen ausreichend viele Angebote vorliegen.

4.1 Polytope stabiler b-Matchings nach Fleiner

Bereits beim Zuweisungsspiel stellen Shapley und Shubik eine Verbindung zur linearen Programmierung her. Mittlerweile existieren einige Arbeiten, wie stabile Matchings in bipartiten Mengen mit einem Präferenzsystem in ein Polytop übergeführt werden können (siehe auch [6]). Dazu werden für ein solches Matching-Modell charakteristische Vektoren ermittelt und deren konvexe Hülle konstruiert. In seiner Arbeit *On the stable b-matching polytope* [7] stellt Fleiner eine Methode vor, mit der dieser Ansatz auf b-Matchings verallgemeinert werden kann. Als Spezialisierung der Definition in Kapitel 2 besitzen alle Kanten die Kapazität 1. Die Vielfachheit b_v eines Knotens v entspricht daher der maximal möglichen Anzahl der zu v inzidenten Kanten.

Sei $G = (U \dot\cup V, E)$ ein endlicher bipartiter Graph. Sei $\mathcal{O} = \{\leq_z | z \in U \cup V\}$ eine Familie linearer Ordnungen, wobei $\leq_z$ eine Ordnung aller zu z möglicherweise inzidenten Kanten ist. Diese Kanten bezeichnen wir auch mit $D(z)$. $(G, \mathcal{O})$ nennen wir ein bipartites Präferenzsystem. $d_M(z)$ ist die Anzahl der zu z inzidenten Kanten bei einem gegebenem Matching M.

Ein stabiles b-Matching ist eine Teilmenge $M \subseteq E$, für die in Anlehnung

an die Stabilitätskriterien beim Hochzeitsproblem oder beim Zuweisungsspiel folgende Kriterien erfüllt sein müssen:

1. Für alle $z \in U \cup V$ muss $d_M(z) \leq b_z$ gelten.
2. M dominiert. Das heißt jede Kante e aus $E \backslash M$ ist inzident zu einem Knoten z, so dass für jede Kante $m \in M$, die inzident zu z ist, $m >_z e$ gilt.

Wir bezeichnen mit $P^b(G, \mathcal{O})$ die konvexe Hülle der charakteristischen Vektoren in $\mathbb{R}^E$ der stabilen b-Matchings eines bipartiten Präferenzsystems $(G, \mathcal{O})$.

Für den Sonderfall $b = 1$ bewies Rothblum [24] die folgende Charakterisierung des Polytops stabiler Matchings:

Satz 4.1.1

Sei $(G, \mathcal{O})$ ein bipartites Präferenzsystem mit $|U| = |V|$ und $E = U \times V$. Dann ist

$$P^1(G, \mathcal{O}) = \left\{ x \in \mathbb{R}^E : x \geq 0, x\left(D(z)\right) \leq 1 \ \forall z \in U \cup V, \ x\left(\phi(e)\right) \geq 1 \ \forall e \in E \right\}$$

mit $\phi(uv) := \{f \in E \ : \ f \geq_u uv$ oder $f \geq_v uv\}$.

In dieser Definition legt $x\left(D(z)\right) \leq 1$ fest, dass das obige Kriterium 1 erfüllt ist. $x\left(\phi(e)\right) \geq 1$ beschreibt das Kriterium 2. Schauen wir uns diesen Ausdruck etwas näher an:

Sei M ein stabiles Matching. Falls dann $e \in M$ gilt, ist der Ausdruck offensichtlich erfüllt. Sei nun $e \in E \backslash M$. Dann muss einer der beiden Endknoten u und v von e inzident zu einer Kante e' sein mit $e' >_u e$ beziehungsweise $e' >_v e$. Denn andernfalls wäre M nicht stabil, weil M nicht dominieren würde. Somit ist auch in diesem Fall die Ungleichung erfüllt.

Fleiner verallgemeinert nun diesen Ansatz von $P^1(G, \mathcal{O})$ auf $P^b(G, \mathcal{O})$. Dazu verwendet er den Vergleichbarkeitssatz von Roth und Sotomayor [22]:

Satz 4.1.2

Seien M und M' zwei stabile b-Matchings eines bipartiten Präferenzsystems $(G, \mathcal{O})$. Sei z ein Knoten in G und $M_z := M \cap D(z)$ und $M'_z := M' \cap D(z)$. Falls $M_z \neq M'_z$ gilt, dann ist $|M_z| = |M'_z| = b_z$ und die b_z $<_z$-besten Kanten von $M_z \cup M'_z$ sind entweder M_z oder M'_z.

Aus diesem Satz lässt sich nun das folgende Korollar ableiten, das im Beweis des darauffolgenden Satzes eine wichtige Rolle spielt:

Korollar 4.1.3

Sei $(G, \mathcal{O})$ ein bipartites Präferenzsystem und $b_z \in \mathbb{N} \ \forall z \in U \cup V$ definiere die Vielfachheit eines jeden Knotens. Dann gibt es für jeden Knoten z in G eine Zerlegung von $D(z)$ in b_z Teilmengen $D_1(z), D_2(z), \ldots, D_{b_z}(z)$, so dass $|M \cap D_i(z)| \leq 1$ für jedes stabile b-Matching M und alle $1 \leq i \leq b_z$ gilt.

Auch wenn mit diesem Korollar lediglich die Existenz einer solchen Partition für ein gegebenes bipartites Präferenzsystem folgt, liefert der Beweis des Korollars eine Möglichkeit, eine solche Partition effizient zu ermitteln. Eine Variation des Algorithmus von Gale und Shapley in [10] kann herangezogen werden, um ein b-Matching M^1 zu finden, das für einen gegebenen Knoten z am wenigsten von allen zulässigen b-Matchings bevorzugt wird. Wird durch dieses Matching z mit weniger als b_z Knoten gematcht, ist man fertig. Ansonsten kann man in höchstens $|D(z)|$ Schritten durch geschicktes Streichen von Kanten zu passenden Matchings und darüber zu einer Partition kommen, die die Aussage des Korollars (4.1.3) erfüllt.

Mit diesen Ergebnissen kann nun als Hauptaussage von [7] ein Polytop stabiler b-Matchings für ein bipartites Präferenzsystem beschrieben werden. Dabei stellt der folgende Satz eine Verallgemeinerung von Satz 4.1.1 dar.

Satz 4.1.4

Sei $(G, \mathcal{O})$ ein bipartites Präferenzsystem und $b_z \in \mathbb{N}\ \forall z \in U \cup V$ definiere die Vielfachheit eines jeden Knotens. Dann gibt es für jeden Knoten z aus G eine Zerlegung von $D(z)$ in Teilmengen $D_1(z), D_2(z), \ldots, D_{b_z}(z)$, so dass gilt

$$\begin{aligned} P^b(G,\mathcal{O}) = \{x \in \mathbb{R}^E :& x \geq 0, \\ & x(D_i(z)) \leq 1\ \forall\ z \in U \cup V, 1 \leq i \leq b_z, \\ & x(\phi_{ij}(uv)) \geq 1\ \forall\ uv \in E, 1 \leq i \leq b_u, 1 \leq j \leq b_v\} \end{aligned}$$

mit

$$\phi_{ij}(uv) := \{uv\} \cup \{uv' : uv' >_u uv, v' \in D_i(u)\} \cup \{u'v : u'v >_v uv, u' \in D_j(v)\}.$$

Beweisidee
Sei die rechte Seite der obigen Gleichung mit R bezeichnet. Die Gleichheit $P^b(G,\mathcal{O}) = R$ wird dadurch bewiesen, dass man $P^b(G,\mathcal{O}) \subseteq R$ und $P^b(G,\mathcal{O}) \supseteq R$ nacheinander beweist.

Die $\subseteq$-Richtung folgt, wenn man als Partition von $D(z)$ diejenige verwendet, die man aus Korollar (4.1.3) ermitteln kann.

Für die $\supseteq$-Richtung wird ein Vektor $x \in R$ als Linearkombination der charakteristischen Vektoren des b-Matchings dargestellt. Nachdem gezeigt wird, dass diese Darstellung möglich ist, folgt daraus auch dieser Teil der Behauptung. □

In Fleiners Ansatz erhält also jeder Agent v eine Vielfachheit b_v. Diese wiederum beschreibt, wie viele Partnerschaften ein Agent gleichzeitig besitzen darf. Das Modell erlaubt jedoch keine multiplen Partnerschaften zwischen zwei Agenten. Außerdem wird das Zuweisungsspiel nicht berücksichtigt. Dies sind auch die wesentlichen Unterschiede zum Modell in Kapitel 5.

4.2 Verschiedene Stabilitätsbegriffe und Substituierbarkeit

Bei den klassischen Matchingproblemen nach Gale und Shapley oder Shapley und Shubik ist der Begriff der Stabilität eines Matchings recht intuitiv zu greifen. Sotomayor zeigt in ihrer Arbeit *Three Remarks on the many-to-many-stable*

matching problem [27], dass es für die Erweiterung klassischer bipartiter Matchingprobleme auf b-Matchings notwendig ist, den Begriff der Stabilität näher zu differenzieren.

Für die folgenden Definitionen seien jeweils zwei disjunkte, endliche Mengen P und Q und Präferenzlisten für jeden Agenten aus $P \cup Q$ wie in den bekannten Modellen gegeben.

Definition 4.2.1

Ein Matching x heißt ***paarweise stabil****, falls es keine Agenten $p \in P$ und $q \in Q$ gibt, die keine Partner sind, die aber eine Partnerschaft untereinander mit eventueller Lösung einer bestehenden Partnerschaft ihren aktuellen Partnerschaften strikt vorziehen würden.*

Definition 4.2.2

Ein Matching x ist ***im Kern des betrachteten Modells*** *(das heißt* ***kernstabil****), falls es keine Teilmenge von $P \cup Q$ gibt, in der alle Agenten dieser Teilmenge neue Partnerschaften bekommen können, die sie gegenüber ihren derzeitigen Parnerschaften echt bevorzugen, indem sie ausschließlich untereinander neue Partnerschaften bilden.*

Definition 4.2.3

Ein Matching x heißt ***mengenstabil****, falls es keine Teilmenge von $P \cup Q$ gibt, so dass alle Agenten dieser Teilmenge dadurch eine strikt bevorzugte Menge an Partnern erhalten können, indem sie neue Partnerschaften innerhalb dieser Teilmenge knüpfen und eventuell einige bestehende Partnerschaften lösen, um ihre Kapazitätsbegrenzung nicht zu überschreiten.*

Dabei stellt Definition (4.2.3) eine Verallgemeinerung der vorangehenden Definitionen dar. Wichtig bei Definition (4.2.3) ist, dass bestehende Partnerschaften mit Agenten außerhalb der betrachteten Menge durchaus aufrecht erhalten werden können. Hier liegt der Unterschied zu den Definitionen (4.2.1) und (4.2.2).

Beim Hochzeitsproblem sind paarweise stabile Matchings gleichzeitig auch kernstabil und mengenstabil. Im Allgemeinen ist die Mengenstabilität jedoch ein stärkeres Stabilitätskriterium als paarweise Stabilität oder Kernstabilität. Für b-Matchings lassen sich Fälle konstruieren, in denen ein paarweise stabiles Matching existiert, das allerdings nicht mengenstabil sein muss.

Außerdem gibt Sotomayor in ihrer Arbeit ein Kriterium an, wann für ein many-to-many-Matching ein paarweise stabiles Matching existiert. Dazu führt sie den Begriff der *Substituierbarkeit von Präferenzen* ein. Seien zwei disjunkte endliche Mengen P und Q mit nicht notwendigerweise strikten Präferenzen gegeben.

Für jedes $y \in P \cup Q$ kann bestimmt werden, welche Teilmengen unter Berücksichtigung der Kapazitätsbeschränkungen für y aus einer Menge A möglicher Partner am meisten bevorzugt werden. Diese Menge sei mit $Ch_y(A)$ bezeichnet. Damit kann folgende Definition erfolgen:

Definition 4.2.4

Sei $y \in P \cup Q$ gegeben und seien F und G zwei disjunkte Mengen möglicher Partner von y. Dann besitzt y substituierbare Präferenzen, wenn gilt:

1. *Für alle $S' \in Ch_y(F \cup G)$ gibt es ein $S \in Ch_y(F)$, so dass $S' \cap F \subseteq S$ und*
2. *Für alle $S \in Ch_y(F)$ gibt es ein $S' \in Ch_y(F \cup G)$, so dass $S' \cap F \subseteq S$*

Mit dieser Definition kann nun die folgende Aussage getroffen werden. In [21] zeigte Roth, dass für ein many-to-many-Matching eine paarweise stabile Lösung existiert, sofern die Präferenzen strikt sind. Diese Aussage wird nun von Sotomayor verallgemeinert:

Satz 4.2.5

Seien P und Q wie oben beschrieben. Jeder Agent $y \in P \cup Q$ besitze substituierbare, aber nicht notwendigerweise strikte Präferenzen und kann mit $b_y > 0$ Agenten eine Partnerschaft bilden. Dann existiert ein paarweise stabiles Matching für dieses Problem.

Die Untersuchungen zur Stabilität bei b-Matchings spielen auch für diese Arbeit eine wichtige Rolle. Denn auch für das dem b-HNS-Algorithmus zu Grunde liegende Modell kann gezeigt werden, dass durch den Algorithmus ein paarweise stabiles Matching erzeugt wird. Der Stabilitätsbegriff wird mit der Einführung von Vielfachheiten deutlich komplexer. Wie bei den Polytopen stabiler b-Matchings nach Fleiner sind im übrigen auch in [27] keine Vielfachheiten zwischen zwei Agenten möglich.

4.3 Das Arbeitsmarktspiel nach Sotomayor

Sotomayor stellt in *A labor market with heterogeneous firms and workers* [29] eine Erweiterung des Zuweisungsspiels von Shapley und Shubik vor, das sie als *Arbeitsmarktspiel* bezeichnet. Ausgangspunkt ist auch hier ein Arbeitsmarkt bestehend aus zwei endlichen, disjunkten Mengen F (*firms*) und W (*workers*). Im Folgenden seien mit $i = 1, \ldots, m$ die Firmen und mit $j = 1, \ldots, n$ die Arbeiter bezeichnet. Während im Modell von Shapley und Shubik jeder Agent höchstens eine Partnerschaft eingehen kann, können in diesem Model jede Firma und jeder Arbeiter mehrere Partnerschaften eingehen. Im wichtigen Unterschied zur vorher vorgestellten Arbeit von Fleiner, ist es in [29] außerdem möglich, dass eine Firma und ein Arbeiter gleich mehrere Kapazitäten in ihre Partnerschaft geben.

Mit ihrer Arbeit zeigt Sotomayor, dass einige Eigenschaften des Zuweisungsspiels von Shapley und Shubik erhalten bleiben, andere wichtige Eigenschaften, die bei den bekannten Algorithmen zur Ermittlung eines stabilen Outcomes verwendet werden, in ihrem verallgemeinerten Modell nicht mehr zutreffen. So besitzt zum Beispiel der Kern dieses Modells, das heißt die Menge der stabilen Outcomes, in der Regel nicht mehr die algebraische Eigenschaft eines Verbandes und hat damit eine entscheidende Eigenschaft im Vergleich zum Zuweisungsspiel

verloren. Daraus folgt allerdings nicht, dass es nicht Fälle gibt, in denen ein optimales stabiles Outcome gefunden werden kann.

Sotomayor bleibt dabei beim Beispiel des Arbeitsmarktes und ordnet jedem Teilnehmer eine individuelle Anzahl an *Arbeitszeiteinheiten* zu. Damit soll ausgedrückt werden, dass jede Firma einen individuellen Bedarf an abzudeckender Arbeitszeit besitzt, während jeder Arbeiter bereit ist, eine bestimmte Anzahl an Zeiteinheiten zu arbeiten. Jede Einheit, die zwischen zwei Agenten $f \in F$ und $w \in W$ vereinbart wird, erwirtschaftet eine Produktivität $c_{fw} \geq 0$. Mit $a = (a_i) \in \mathbb{R}^m$ und $b = (b_j) \in \mathbb{R}^n$ werden die zur Verfügung stehenden Arbeitszeiteinheiten der Firmen und der Arbeiter definiert. $c = (c_{ij}) \in \mathbb{R}^{m \times n}$ ist die Matrix, in den die Produktivitäten aller möglichen Partnerschaften definiert sind. Damit wird der Arbeitsmarkt M über das 5-Tupel $M = (F, W, c, a, b)$ definiert. Mit x_{ij} bezeichnen wir die Anzahl der Arbeitszeiteinheiten, die eine Firma i und ein Arbeiter j miteinander vereinbaren. Die sich daraus ergebende Matrix $x \in \mathbb{R}^{m \times n}$ nennen wir *Arbeitsverteilung.*

Definition 4.3.1

Die Arbeitsverteilung x heißt zulässig, falls

$$\sum_{j' \in W} x_{ij'} \leq a_i \text{ für alle } i \in F \tag{4.1}$$

und

$$\sum_{i' \in F} x_{i'j} \leq b_j \text{ für alle } j \in W \tag{4.2}$$

gilt.

Die Arbeitsverteilung kann man als ein spezielles Matching in einem bipartiten Graphen interpretieren. Es ist klar, dass in Analogie zu den bislang vorgestellten Problemen eine Arbeitsverteilung nur dann zulässig sein kann, wenn die zur Verfügung stehenden Arbeitszeitseinheiten eines jeden Agenten nicht überschritten werden. Ähnlich anschaulich ist die Definition eines optimalen Matchings in diesem Modell.

Definition 4.3.2

Eine zulässige Arbeitsverteilung x heißt optimal, wenn

$$\sum_{i \in F} \sum_{j \in W} c_{ij} x_{ij} \geq \sum_{i \in F} \sum_{j \in W} c_{ij} x'_{ij}$$

für alle zulässigen Arbeitsverteilungen x' gilt.

Neben der Arbeitsverteilung ist auch die Geldverteilung (Payoff) relevant. Nachdem ein Agent aus mehreren Koalitionen Profit erzielen kann, muss die Geldverteilung allgemeiner definiert werden als beim klassischen Zuweisungsspiel. Definiere für alle $R \subseteq F$ und $S \subseteq W$ das Payoff $P(R \cup S)$ der Koalition $R \cup S$ als das Maximum aus $\sum_{R \times S} c_{ij} x_{ij}$ für alle zulässigen Arbeitsverteilungen x. Damit

ist eine Arbeitsverteilung x genau dann optimal, wenn $\sum_{F\times W} c_{ij}x_{ij} = P(F \cup W)$ ist. Für die folgende Definition legen wir noch die Schreibweisen $u(R) = \sum_R u_i$ und $v(S) = \sum_S v_j$ fest.

Definition 4.3.3

Eine Geldverteilung (Payoff) ist ein Vektor $(u, v) \in \mathbb{R}^m \times \mathbb{R}^n$. Sie heißt zulässig, falls für alle $(i, j) \in F \times W$ gilt:

$$u(F) + v(W) \leq P(F \cup W) \text{ und}$$
$$u_i \geq 0, v_j \geq 0.$$

Die Stabilität eines Payoffs lässt sich allerdings nicht so einfach vom klassischen Zuweisungsspiel übertragen. Dort besteht Stabilität, wenn es nicht für beide Agenten echt profitabel ist, ihre eventuell bestehende Koalition aufzugeben, um eine neue miteinander zu begründen. Sotomayor zeigt beispielhaft, dass diese paarweise Stabilität für ihr Arbeitsmarktspiel nicht angemessen sein muss. Instabilitäten können durch jede beliebig große Teilmenge aus Agenten entstehen.

Aus diesem Grund führt sie das Konzept des Kerns, das Shapley und Shubik für das Zuweisungsspiel eingeführt haben, für das Arbeitsmarktspiel fort.

Definition 4.3.4

Der Kern des Arbeitsmarktspiels besteht aus allen stabilen Payoffs. Ein zulässiges Payoff (u, v) ist damit genau dann im Kern enthalten, wenn es von keinem anderen zulässigen Payoff (u', v') dominiert wird. Dabei dominiert (u', v') (u, v), wenn es Teilmengen $R \subseteq F$ und $S \subseteq W$ gibt, so dass jeder Agent aus $R \cup S$ das Payoff (u', v') strikt gegenüber (u, v) bevorzugt.

Ein Vergleich zwischen Zuweisungsspiel und Arbeitsmarktspiel zeigt nun einige Parallelen auf. So folgt aus einer optimalen Arbeitsverteilung x, dass diese mit jedem stabilen Payoff verträglich ist. Außerdem gilt die wichtige Eigenschaft, dass der Kern des Arbeitsmarktspiels nicht leer ist.

Auf der anderen Seite gibt es auch signifikante Unterschiede. Während beim Zuweisungsspiel der Kern mit den Lösungen des zugehörigen dualen Problems übereinstimmt, muss dies für das Arbeitsmarktspiel nicht zutreffen. Auch Aussagen wie „Ein für die Firmen optimales Payoff wird von den Arbeitern am wenigsten bevorzugt und umgekehrt.“ treffen im Allgemeinen nicht mehr zu. Außerdem ist der Kern des Problems kein algebraischer Verband mehr. Und aus den Stabilitätsüberlegungen von Sotomayor folgt auch, dass paarweise Statbilität und Mengenstabilität nicht zusammenfallen müssen.

Diese Unterschiede führen dazu, dass die bekannten Lösungsalgorithmen für das Zuweisungsspiel nicht einfach auf das Arbeitsmarktspiel übertragen werden können.

Das Arbeitsmarktspiel ist sehr ähnlich zu dem in dieser Arbeit in Kapitel 5 vorgestellten Modell. Insbesondere sind über den Begriff der Arbeitszeiteinhei-

ten mehrfache Partnerschaften zwischen zwei Agenten möglich. In meinem Modell werden in Erweiterung zum Arbeitsmarktspiel auch Koalitionen wie beim Hochzeitsproblem zugelassen. Außerdem sind in [29] die Kapazitäten der Kanten nicht beschränkt.

4.4 Diskrete konkave Nutzenfunktionen

Im vorangegangenen Abschnitt wurde eine Erweiterung des Zuweisungsspiels auf b-Matchings vorgestellt. Eriksson und Karlander führen das Hochzeitsproblem und das Zuweisungsspiel zusammen. In ihrer Arbeit *A General Two-Sided Matching Market with Discrete Concave Utility Functions*[8] verallgemeinern Fujishige und Tamura diese beiden Modelle.

In ihrem Modell kann jeder Agent mit mehreren Agenten der Gegenseite eine Partnerschaft knüpfen. Außerdem können zwei Partner auch eine Vielfachheit ihrer Partnerschaft vereinbaren. Hier deckt sich dieses Modell mit dem von Sotomayor. Die Präferenzen aller Agenten werden als sogenannte *diskrete konkave Nutzenfunktionen* ausgedrückt. Solche Funktionen werden auch $M^\natural$-konkave Funktionen genannt.[1] Dieser Begriff soll nun beschrieben werden.

Sei $E \neq \emptyset$ eine endliche Menge.[2] Für jedes $x \in \mathbb{Z}^E$ wird der positive Träger $\mathrm{supp}^+(x)$ und der negative Träger $\mathrm{supp}^-(x)$ definiert durch

$$\mathrm{supp}^+(x) = \{e \in E | x(e) > 0\}, \quad \mathrm{supp}^-(x) = \{e \in E | x(e) < 0\}$$

Außerdem seien für $x, y \in \mathbb{Z}^E$

$$(x \wedge y)(e) := \min\{x(e), y(e)\} \quad \text{und} \quad (x \vee y)(e) := \max\{x(e), y(e)\}.$$

Für $x \in \mathbb{Z}^E$, $p \in \mathbb{R}^E$ und eine Funktion $f : \mathbb{Z}^E \to \mathbb{R} \cup \{-\infty\}$ definieren wir außerdem

$$\langle p, x \rangle := \sum_{e \in E} p(e)x(e) \text{ und}$$
$$f\,[p]\,(x) := f(x) + \langle p, x \rangle .$$

Die Menge der maximierenden Argumente von f über einer Teilmenge $U \subseteq \mathbb{Z}^E$ und den wirksamen Bereich von f definieren wir schließlich noch über

$$\arg\max\{f(y) | y \in U\} := \{x \in U | f(x) \geq f(y) \text{ für alle } y \in U\}$$
$$\mathrm{dom} f := \{x \in \mathbb{Z}^E | f(x) > -\infty\}.$$

Definition 4.4.1

Eine Funktion $f : \mathbb{Z}^E \to \mathbb{R} \cup \{-\infty\}$ *mit* $\mathrm{dom} f \neq \emptyset$ heißt $M^\natural$-konkav, wenn für beliebige $x, y \in \mathrm{dom} f$ *und ein beliebiges* $e \in \mathrm{supp}^+(x - y)$ *ein* $e' \in \mathrm{supp}^-(x - y) \cup \{0\}$ *existiert, so dass gilt*

$$f(x) + f(y) \leq f(x - \chi_e + \chi_{e'}) + f(y + \chi_e - \chi_{e'}).$$

[1] Das hochgestellte $\natural$ wird als „natural“ gelesen.

[2] Hier wird der allgemeine Fall vorgestellt. Später wird $E = P \times Q$ die Kantenmenge eines Graphen bezeichnen.

Eine $M^\natural$-konkave Funktion besitzt eine konkave Erweiterung auf $\mathbb{R}^E$. Außerdem impliziert $M^\natural$-Konkavität Substituierbarkeit. Daraus folgt aber mit [27] die wichtige Feststellung, dass ein b-Matching-Problem, dessen Präferenzliste als $M^\natural$-konkave Funktion dargestellt werden kann, ein paarweise stabiles Matching besitzt.

Unter anderem am Beispiel des Hochzeitsproblems zeigen Fujishige und Tamura, wie aus einer Präferenzliste eine $M^\natural$-Funktion erzeugt werden kann. Seien dazu der Graph $G = (M \dot\cup W, E := M \times W)$ mit $|M| = |W| = n$ und die Präferenzmatrizen $A = (a_{ij}), B = (b_{ij}) \in \mathbb{R}^{n \times n}$ gegeben. Ergänzend zu den bekannten Präferenzeigenschaften im Hochzeitsmodell nennen wir die Partnerschaft (i, j) aus Sicht von i beziehungsweise j *akzeptierbar*, wenn $a_{ij} > 0$ beziehungsweise $b_{ij} > 0$ gilt. Nicht akzeptierbare Partnerschaften sollen immer den Wert $a_{ij} = -\infty$ beziehungsweise $b_{ij} = -\infty$ annehmen. Der Matchingvektor sei mit $x = (x_{ij}) \in \{0, 1\}^E$ bezeichnet. Damit können nun zwei Nutzenfunktionen f_M für M und f_W für W definiert werden:

$$f_M(x) := \begin{cases} \sum\limits_{(i,j)\in E} a_{ij}x_{ij} & \text{falls } x \in \{0,1\}^E \text{ und } \sum\limits_{j\in W} x_{ij} \leq 1 \;\forall\, i \in M \\ -\infty & \text{sonst} \end{cases} \tag{4.3}$$

$$f_W(x) := \begin{cases} \sum\limits_{(i,j)\in E} b_{ij}x_{ij} & \text{falls } x \in \{0,1\}^E \text{ und } \sum\limits_{i\in M} x_{ij} \leq 1 \;\forall\, j \in W \\ -\infty & \text{sonst} \end{cases} \tag{4.4}$$

Fujishige und Tamura zeigen, dass die beiden Funktionen $M^\natural$-konkav sind.

Die Eigenschaften eines (paarweise) stabilen Matchings im Hochzeitsproblem können nun über die Nutzenfunktionen f_M (4.3) und f_W (4.4) ausgedrückt werden. Damit heißt ein Matchingvektor x paarweise stabil im Hochzeitsproblem, falls Vektoren $z_M, z_W \in \{0, 1\}^E$ existieren, so dass gilt:

$$1 = z_M \vee z_W \tag{4.5}$$

$$x \text{ maximiert } f_M \text{ in } \{y \in \mathbb{Z}^E | y \leq z_M\} \tag{4.6}$$

$$x \text{ maximiert } f_W \text{ in } \{y \in \mathbb{Z}^E | y \leq z_W\} \tag{4.7}$$

Ein Vektor $x \in \{0, 1\}^E$, der (4.6) und (4.7) erfüllt, muss zu einem Matching gehören, denn $x \in \mathrm{dom} f_M \cap \mathrm{dom} f_W$. Aussage (4.6) besagt dabei, dass jedes Element aus M einen möglichst guten Partner aus W besitzt. Die entsprechende Aussage für Elemente aus W folgt aus (4.7). (4.5) stellt sicher, dass kein Element ungematcht ist und dass kein untereinander nicht gematchtes Paar existiert, das eine Partnerschaft untereinander gegenüber den aktuellen Partnerschaften bevorzugen würde. Somit gehört x zu einem paarweise stabilen Matching.

Sei umgekehrt x ein aus einem paarweise stabilen Matching resultierender Vektor in $\{0, 1\}^E$. Dann kann man z_M konstruieren. Setze dafür $z_M(i, j) = 0$ für alle $(i, j) \in E$ für die i j gegenüber seinem aktuellen Partner bevorzugt oder bei denen i im aktuellen Matching ungematcht ist. Setze andernfalls $z_M(i, j) = 1$. Analog kann z_W konstruiert werden. Diese Vektoren erfüllen (4.5) bis (4.7).

Auch die Eigenschaften des Zuweisungsspiels können über Nutzenfunktionen dargestellt werden. Für die Herleitung wird auf [8] verwiesen. Dabei bleiben die Nutzenfunktionen wie in (4.3) und (4.4) bestehen. Ein Matching x ist genau dann ein paarweise stabiles Matching im Zuweisungsspiel, wenn ein $p \in \mathbb{R}^E$

existiert, so dass die beiden folgenden Aussagen erfüllt sind

$$x \text{ maximiert } f_M[+p] \tag{4.8}$$
$$x \text{ maximiert } f_W[-p] \tag{4.9}$$

Nach dieser Einführung über $M^\natural$-konkave Funktionen soll nun das Modell von Fujishige und Tamura vorgestellt werden. Seien dazu M und W zwei endliche, disjunkte Mengen, sei $E := M \times W$ und seien $f_M, f_W : \mathbb{Z}^E \to \mathbb{R} \cup \{-\infty\}$ zwei $M^\natural$-konkave Funktionen. Über f_M und f_W werden also die Nutzen der Agenten aus M beziehungsweise W aggregiert. E wird in zwei Mengen F und R unterteilt. F beschreibt dabei die Menge der flexiblen Elemente und R die Menge der rigiden Elemente und es gilt $E = F \dot\cup R$. Für f_M und f_W nehmen wir weiterhin an, dass $\mathrm{dom} f_M$ und $\mathrm{dom} f_W$ beschränkt und vererbbar sind und 0 als gemeinsames minimales Element besitzen.

Dabei bedeutet Vererbbarkeit, dass aus $0 \leq x_1 \leq x_2 \in \mathrm{dom} f_M$ (beziehungsweise $\mathrm{dom} f_W$) folgt, dass $x_1 \in \mathrm{dom} f_M$ (beziehungsweise $\mathrm{dom} f_W$). Nachdem x_{ij} die Vielfachheit einer Partnerschaft zwischen i und j beschreibt, bedeutet diese Eigenschaft, dass ein Agent die Vielfachheit einer Partnerschaft ohne Zustimmung seines Partners reduzieren kann. Diese Eigenschaft besteht in den klassischen Matchingmodellen ganz natürlich.

Sei außerdem noch $z \in \mathbb{N}^E$ ein Vektor aus natürlichen Zahlen, für den

$$\mathrm{dom} f_M \cup \mathrm{dom} f_W \subseteq \left\{ y \in \mathbb{Z}^E | 0 \leq y \leq z \right\}$$

gilt.

Definition 4.4.2

$x \in \mathrm{dom} f_M \cap \mathrm{dom} f_W$ heißt eine $f_M f_W$-paarweise-stabile Lösung (unter Berücksichtigung von (F, R)), wenn $p \in \mathbb{R}^E$ sowie $z_M, z_W \in \mathbb{Z}^E$ existieren, so dass die folgenden Bedingungen erfüllt sind:

$$p|_R = 0, \tag{4.10}$$
$$z|_R = z_M \vee z_W, \tag{4.11}$$
$$x \in \arg\max \{ f_M[+p](y) | y|_R \leq z_M \} \tag{4.12}$$
$$x \in \arg\max \{ f_W[-p](y) | y|_R \leq z_W \} \tag{4.13}$$

Die Bedingung (4.10) stellt hierbei sicher, dass bei rigiden Kanten keine side payments erfolgen. In der Bedingung (4.11) wird der Vektor 1 in (4.5) durch $z|_R$ ersetzt. z beschreibt die den einzelnen Agenten jeweils zur Verfügung stehenden Arbeitszeitkapazitäten. Insbesondere folgen also aus den Kriterien (4.11) bis (4.13) die Kriterien (4.5) bis (4.9) in den eingangs betrachteten Sonderfällen.

Die Hauptaussage der Arbeit von Fujishige und Tamura liefert nun der folgende Satz:

Satz 4.4.3

Seien $f_M, f_W : \mathbb{Z}^E \to \mathbb{R} \cup \{-\infty\}$ $M^\natural$-konkave Funktionen, für die $\mathrm{dom} f_M$ und $\mathrm{dom} f_W$ beschränkt und vererbbar sind und 0 als gemeinsames minimales Element besitzen. Sei $F \dot\cup R = E$ eine beliebige Zerlegung von E.

Dann existiert ein $f_M f_W$-paarweise-stabile Lösung unter Berücksichtigung von (F, R).

Dabei liefern Fujishige und Tamura im Beweis gleich einen Algorithmus, mit dem ein solches $f_M f_W$-paarweise-stabiles Matching konstruiert werden kann. Dieser Algorithmus liefert ein $p \in \mathbb{R}^E$, $x_M, x_W \in \mathbb{Z}^E$ und $z_M, z_W \in \mathbb{Z}^R$, so dass die Bedingungen (4.10) bis (4.13) erfüllt werden, wobei in (4.12) x durch x_M und in (4.13) x durch x_W ersetzt wird. Zum Ende des Algorithmus muss dann ergänzend noch gelten

$$x_M = x_W \tag{4.14}$$

Der Algorithmus besteht aus zwei Phasen. Phase 1 liefert dabei ein Outcome, das die Bedingungen (4.10) bis (4.13) erfüllt. Darüberhinaus gilt auch $x_M|_R = x_W|_R$ und $x_M \leq x_W$. Damit ist klar, dass dieser Algorithmus durchaus nach Phase 1 bereits enden kann. In Phase 2 werden die Ergebnisse von Phase 1 als Input herangezogen und Teile der Phase 1 mit vertauschten Rollen von M und W erneut durchgeführt. Da damit nach Abschluss von Phase 2 insbesondere $x_W \leq x_M$ gilt, wird nun auch die Bedingung (4.14) erfüllt.

Zu Beginn des Algorithmus werden nun $z_M := z|_R$, $z_W := 0$ und $p := 0$ gesetzt. Erste x_M und x_W können daraus konstruiert werden. Zwei Unteralgorithmen sorgen dafür, dass die Inputdaten so modifiziert werden, dass die genannten Bedingungen nach Abschluss von Phase 1 erfüllt werden. Phase 2 unterscheidet sich grundsätzlich von Phase 1 nur dadurch, dass ein anderer Input herangezogen wird. Wie bereits beschrieben, wird das Outcome der Phase 1 in Form von z_M, z_W, x_M, x_W und p als neuer Input verwendet, wobei M und W vertauscht werden.

Es ist auch recht einfach, zum Arbeitsmarktspiel von Sotomayor $M^\natural$-Funktionen zu konstruieren und damit zu zeigen, dass das Modell von Fujishige und Tamura eine Verallgemeinerung des Arbeitsmarktspiels darstellt. Die Bedingungen (4.1) und (4.2) werden dazu wie folgt berücksichtigt:

$$f_M(x) := \begin{cases} \sum\limits_{(i,j)\in E} c_{ij}x_{ij} & \text{falls } x \in \mathbb{Z}^E \text{ 4.1 erfüllt und } x \geq 0 \\ -\infty & \text{sonst} \end{cases}$$

$$f_W(x) := \begin{cases} 0 & \text{falls } x \in \mathbb{Z}^E \text{ 4.2 erfüllt und } x \geq 0 \\ -\infty & \text{sonst} \end{cases}$$

In *A Two-Sided Discrete-Concave Market with Possibly Bounded Side Payments: An Approach by Discrete Convex Analysis* [9] verallgemeinern Fujishige und Tamura das eben vorgestellte Modell noch. Für jede mögliche Partnerschaft zwischen zwei Agenten wird festgelegt, in welchem Bereich sich side payments bewegen dürfen. Dadurch wird die in [8] festgelegte Eigenschaft, ob eine Kante rigide oder flexibel ist, aufgeweicht. Stattdessen kann die Höhe möglicher side payments festgelegt werden. Im Extremfall, dass keine side payments erlaubt sind, erhält man wieder die Eigenschaft einer rigiden Kante. Im anderen Extremfall vollkommener Freiheit bei der Aufteilung des Profits einer Partnerschaft erhält man eine flexible Kante. Entsprechend nennen Fujishige und Tamura dieses Modell auch ein *Zuweisungsspiel mit möglicherweise beschränkten side payments.*

Die Modelle von Fujishige und Tamura verallgemeinern viele der bislang angesprochenen Modelle. Dabei ist die Modellentwicklung in vielen Teilen vergleichbar mit dem in dieser Arbeit vorgestellten Modell. Fujishige und Tamura teilen in [8] die Kanten in Analogie zu Eriksson und Karlander zu Beginn in flexible und rigide Kanten auf. Nachdem mein Modell auf dem von Hochstättler, Nickel und Schiess basiert, steht wie dort nicht von vornherein fest, ob eine Kante zwischen zwei Agenten rigide oder flexibel ist. Es kann sogar passieren, dass eine rigide und eine flexible Kante gleichzeitig zwischen zwei Agenten existiert.

4.5 Bipartite b-Matchings und Wahrscheinlichkeitsverteilungen

Angerissen werden soll an dieser Stelle auch noch ein im Vergleich zu den bisherigen Modellen völlig anderer Ansatz. Die Zielfunktion eines gewichteten bipartiten b-Matchingproblems wird in *Loopy Belief Propagation for Bipartite Maximum Weight b-Matching* [14] als eine Wahrscheinlichkeitsverteilungsfunktion formuliert. Ausgangspunkt für diesen Ansatz sind computertechnische Verfahren aus den Bereichen der künstlichen Intelligenz und des maschinellen Lernens.

Unter *Belief Propagation* ist allgemein gesprochen ein iterativer Algorithmus zu verstehen, mit dem probabilistische Rückschlüsse auf Grenzwerte gezogen werden sollen. Belief Propagation konvergiert im allgemeinen nicht und ist daher aus rein mathematischer Sicht oft nicht zielführend, während approximierte Werte für die Anwendung im Computerbereich häufig ausreichend sind. Belief Propagation terminiert vor allem dann oft nicht, wenn der zu Grunde liegende Graph Kreise beinhaltet, woraus sich der Ausdruck *loopy* begründet. In ihrer Arbeit stellen Huang und Jebara einen Sonderfall vor. Belief Propagation wird dabei für ein b-Matchingproblem eingesetzt, welches mit diesem Algorithmus effizient und nachweislich exakt gelöst werden kann.

Sei dazu ein bipartiter Graph $G = (U \dot\cup V, E)$ mit $U = \{u_1, \ldots, u_n\}$ und $V = \{v_1, \ldots, v_n\}$ gegeben. Sei $A := (a_{ij}) \in \mathbb{R}^{n \times n}$ die Gewichtsmatrix von G. a_{ij} ist also das Gewicht der Kante (u_i, v_j). Ein b-Matching wird in [14] definiert als Funktion $M(u_i)$ oder $M(v_j)$, die für jedes gegebene b-Matching die Knoten zurückliefert, die adjazent zum Knoten im Argument der jeweiligen Funktion sind. Dabei ist die Knotenvielfachheit $b \in \mathbb{Z}^+$ für alle Knoten aus G gleich. Außerdem besitzt jede Kante die Kapazität 1.

Damit lässt sich das Problem, ein maximal gewichtetes b-Matching zu finden, wie folgt beschreiben:

$$\max_M \mathcal{W}(M) = \max_M \sum_{i=1}^{n} \sum_{v_k \in M(u_i)} a_{ik} + \sum_{j=1}^{n} \sum_{u_l \in M(v_j)} a_{lj}$$

$$\text{unter den Nebenbedingungen} \quad |M(u_i)| = b \ \forall i \in \{1, \ldots, n\}$$
$$|M(v_j)| = b \ \forall j \in \{1, \ldots, n\}$$

Wir können annehmen, dass $b \leq n$ gilt, andernfalls besitzt das zugrunde liegende Modell kein b-Matching. Wir definieren mit X beziehungsweise Y die Menge aller b-elementigen Teilmengen von V beziehungsweise U. Damit gibt es für

jedes $u_i \in U$ ein $x_i \in X$ und für jedes $v_j \in V$ ein $y_j \in Y$, so dass $x_i = M(u_i)$ und $y_j = M(v_j)$ gilt. Damit werden nun Funktionen definiert.

$$\phi(x_i) = \exp\left(\sum_{v_j \in x_i} a_{ij}\right), \qquad \phi(y_j) = \exp\left(\sum_{u_i \in y_j} a_{ij}\right) \tag{4.15}$$

$$\varphi(x_i, y_j) = \neg(v_j \in x_i \oplus u_i \in y_j) \tag{4.16}$$

Dabei nimmt die Gleichung (4.16) als logische, binäre Funktion die Werte 0 oder 1 an. Sie ist genau dann 0, wenn entweder das zu y_j gehörende v_j in x_i enthalten ist, das zu x_i gehörende u_i aber nicht in y_j oder wenn das zu x_i gehörende u_i in y_j, das y_j gehörende v_j aber nicht in x_i enthalten ist.

Mit diesen Definitionen lässt sich nun die Zielfunktion eines wie oben festgelegten gewichteten b-Matchings auch wie folgt schreiben:

$$p(X, Y) = \frac{1}{Z} \prod_{i=1}^{n} \prod_{i=1}^{n} \varphi(x_i, y_j) \prod_{k=1}^{n} \phi(x_k)\phi(y_k) \tag{4.17}$$

Dabei stellt Z die Normalisierungskonstante dar, die dafür sorgt, dass $p(X, Y)$ eine Wahrscheinlichkeitsdichtefunktion ist, indem sie die „Fläche unter der Funktion“ zu 1 normiert.

Diese Wahrscheinlichkeitsfunktion kann nun mittels des sogenannten *max-product Algorithmus* maximiert werden. Für eine Einführung in diesen Algorithmus sei auf *Factor Graphs and the Sum-Product Algorithm* [17] und vor allem auf *Maximum Weight Matching via Max-Product Belief Propagation* [2] verwiesen. Grundsätzlich geht es bei dem max-product Algorithmus darum, dass Nachrichten (*messages*, [14]) in Form von Vektoren zwischen beliebigen x_i und y_j ausgetauscht werden und Abschätzungen für die maximalen Randverteilungen gespeichert werden.

Der hier vorgestellte Algorithmus weist dabei zwei Besonderheiten auf. Zum einen terminiert er, obwohl in einem bipartiten Graphen in der Regel Kreise vorliegen, zum anderen kann er effizient in einer Laufzeit von $\mathcal{O}(bn^3)$ durchgeführt werden. Dies gelingt dadurch, dass bei der Übertragung der Nachrichten vermieden werden kann, dass alle $\binom{n}{b}$ Elemente der Vektor-Nachricht formuliert werden müssen. Das Problem wird auf die Beschreibung eines b-dimensionalen Hyperkubus reduziert.

Dieses Modell wurde insbesondere deswegen vorgestellt, um die Anwendungsmöglichkeit bipartiter b-Matchings auch in anderen Bereichen der Mathematik und in anderen Wissenschaftsfeldern aufzuzeigen.

4.6 Übersicht

Eine schöne Übersicht über die Zusammenhänge der klassischen Matchingspiele und und deren Erweiterung auf b-Matchings findet sich in [8]. Eine leichte Variation hiervon stellt das Diagramm 4.1 dar, das die bislang vorgestellten Arbeiten in Zusammenhang bringt.

Fleiner baut sein Modell, das zum stabilen b-Matching Polytop führt, ausgehend vom Hochzeitsproblem und vom Collegezuweisungsproblem auf. Dass der klassische HNS-Algorithmus eine Verallgemeinerung des Modells von Eriksson

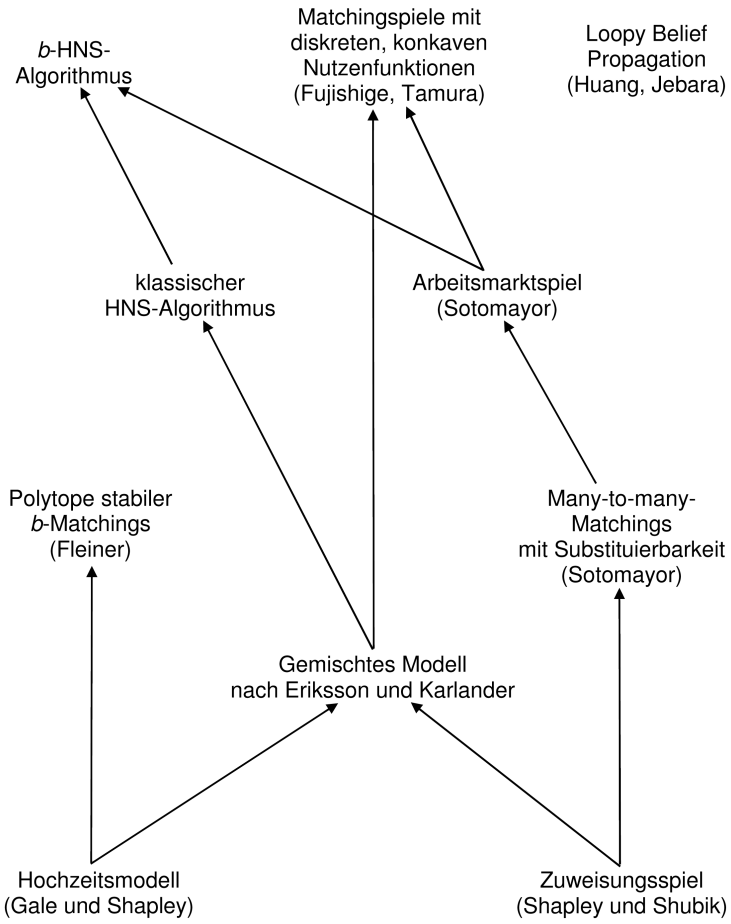

Abbildung 4.1: Zusammenhänge zwischen verschiedenen Modellen zweiseitiger Matching-Märkte.

und Karlander darstellt, zeigen Hochstättler, Nickel und Schiess in [13]. Dass der b-HNS-Algorithmus wiederum eine Verallgemeinerung des klassischen HNS-Algorithmus, aber auch des Arbeitsmarktmodells von Sotomayor ist, wird in dieser Arbeit in den Abschnitten 5.7 und 5.8 gezeigt. Nachdem [14] auf keinem der klassischen Matchingprobleme beruht, wird diese Arbeit ohne Verbindung zu den anderen Modellen dargestellt.

Kapitel 5

Erweiterung des HNS-Modells

5.1 Beschreibung

In Weiterführung der Arbeit von Hochstättler, Nickel und Schiess [13] soll nun ein Modell vorgestellt werden, in dem auch b-Matchings möglich sind. Mit diesen Vielfachheiten an den einzelnen Knoten wird das zu Beginn dieser Arbeit vorgestellte Problem modelliert: Jede Firma besitzt mehrere Kapazitäten, die man in Anlehnung an das Arbeitsmarktspiel von Sotomayor als Arbeitszeiteinheiten interpretieren kann. Entsprechend soll für jeden Arbeiter festgelegt werden, wie viele Arbeitszeiteinheiten er bereit ist zu arbeiten.

Sei dazu $G = (P \dot\cup Q)$ ein bipartiter Graph. Die Präferenzmatrizen $A, B, C \in \mathbb{R}_+^{n \times m}$ erhalten die gleiche Bedeutung wie in [13]. A und B beschreiben also die Präferenzmatrizen bei rigiden Partnerschaften und C die Profite bei flexiblen Partnerschaften. Darüber hinaus definieren wir für jeden Knoten v eine Vielfachheit b_v. Dies geschieht über die beiden Vektoren P_{cap} und Q_{cap}, deren Elemente aus nichtnegativen ganzen Zahlen bestehen und die zur Verfügung stehenden Arbeitszeiteinheiten jedes Agenten beschreiben.

Bis zu dieser Stelle lässt sich das Problem noch trivial mit den klassischen HNS-Algorithmus lösen - wenn auch in der Regel nicht besonders schnell: Man ersetze jeden einzelnen Knoten v durch b_v Knoten und wende auf den so erhaltenen Graphen den klassischen HNS-Algorithmus an. Nach dessen Abschluss fügt man die vormals getrennten Knoten wieder zu einem einzigen zusammen. So erhält man offensichtlich ein stabiles b-Matching.

Mit dem b-HNS-Algorithmus soll aber ein effizienterer Algorithmus vorgestellt werden. Insbesondere hängt die Laufzeit dieses neuen Algorithmus, wie wir später sehen werden, nur logarithmisch von den Vielfachheiten auf den einzelnen Knoten ab, was in diesem trivialen Ansatz offensichtlich nicht der Fall ist. Außerdem berücksichtigen wir die Tatsache, dass Arbeitsverträge auf eine bestimmte Stundenzahl begrenzt sind. Auf dem Arbeitsmarkt führen auch tarifvertragliche oder steuerliche Aspekte („400-Euro-Jobs“) zu solchen Begrenzungen. Im vorgestellten Modell kann dabei für jeden möglichen Arbeitsvertrag separat festgelegt werden, wie hoch diese Begrenzung ist. Es ist erlaubt, dass zwischen zwei Agenten sowohl eine rigide als auch eine flexible Kante besteht.

Durch diese Begrenzungen der Kantenkapazitäten taucht ein Problem auf, das beim klassischen HNS-Algorithmus noch keine Rolle gespielt hat. Es ist nicht

sichergestellt, dass zum Abschluss des Algorithmus alle Kapazitäten eines jeden Agenten gematcht sind. Dies gilt auch dann, wenn die Summe der Kapazitäten aller zu einem Agenten v inzidenten Kanten größer als b_v ist und selbst dann nicht zwingend, wenn seine möglichen Vertragspartner in Summe ausreichend viele Kapazitäten anbieten.

Im Outcome muss berücksichtigt werden, ob zwischen zwei Agenten eine flexible oder eine rigide Kante besteht - wobei sich beides nicht ausschließt - , wie viele Kapazitäten in jeder Kante stecken und welchen Profit jeder Agent aus seinen Koalitionen bezieht.

Damit taucht die nächste Besonderheit auf. Den Profit bezieht ein Agent aus den bestehenden Kanten. Diese werden aber in der Regel nicht den gleichen Profit ergeben. Man stelle sich folgenden einfachen Fall vor: P und Q besitzen je zwei Elemente mit einem Bedarf von jeweils 2. Jede der vier möglichen rigiden Kanten hat die Kapazität 1, die flexiblen Kanten haben jeweils keine Kapazität. Nachdem jeder Profit nichtnegativ ist, ist es auf jeden Fall für jeden Agenten vorteilhaft, im Sinne der Profitmaximierung alle Kanten voll auszuschöpfen. Das führt aber eben in der Regel dazu, dass jeder Agent zwei unterschiedliche Profite aus den beiden Kanten erhält.

Daher werden die bekannten Payoffvektoren (u, v) durch je zwei Matrizen für rigide und flexible Profite ersetzt. Wir bezeichnen sie mit U^R, V^R, U^F und V^R.

Die Zulässigkeit eines Outcomes ist recht leicht zu überlegen. Zum einen müssen die Kapazitätsbegrenzungen aller Knoten und Kanten berücksichtigt werden. Falls i und j rigide gematcht sind, dann müssen die Einträge an der Stelle i, j in U^R und A sowie in V^R und B übereinstimmen. Falls i und j flexibel gematcht sind, muss die Summe der beiden Einträge an der Stelle i, j in U^F und V^F gleich dem Eintrag an der Stelle i, j in C sein.

Wann ist nun ein solches Outcome stabil? Wir überlegen uns dies in Anlehnung an die bekannten Matching-Probleme und unter besonderer Beachtung der Definition paarweiser Stabilität in 4.2.1. Seien $i \in P$ und $j \in Q$ zwei beliebige Agenten. Wir bezeichnen mit $(i, j)^F$ die flexible und mit $(i, j)^R$ die rigide Kante zwischen i und j. Ein Outcome in diesem Modell ist paarweise stabil, wenn immer gilt

1. i und j können keine weiteren Kapazitäten in die Kanten $(i, j)^F$ und $(i, j)^R$ geben oder

2. jede Verschiebung einer Kantenkapazität von einer bestehenden Kante nach $(i, j)^F$ oder $(i, j)^R$ ist nicht für beide Agenten i und j mit einer Steigerung des Gesamtprofits verbunden.

Dabei kann Fall 1 eintreten, wenn einer der beiden Agenten bereits alle Kapazitäten in den beiden Kanten $(i, j)^F$ und $(i, j)^R$ untergebracht hat. Fall 1 kann auch dann auftreten, wenn die beiden Kanten zwischen i und j bereits völlig ausgelastet sind.

Wenn in einem paarweise stabilen Outcome beide Agenten noch Kapazitäten außerhalb dieser Kanten besitzen, dann sind diese mit einem nichtnegativen Profit belegt. Der zu erwartende Profit aus einer Kapazitätsverschiebung nach $(i, j)^F$ oder $(i, j)^R$ führt dann aber nicht zu einer strikten Verbesserung des Gesamtprofits beider Agenten.

Bei diesen Betrachtungen muss auch der Fall berücksichtigt werden, dass es den beiden Agenten möglich sein kann, Kapazitäten zwischen der rigiden und der flexiblen Kante zu verschieben. Auch hier muss einer der beiden genannten Gründe gegen eine Kapazitätsverschiebung sprechen.

5.2 Das Modell

An dieser Stelle wird das mathematische Modell formuliert. Auf Basis dieses Modells wird im nächsten Kapitel der Algorithmus entwickelt.

5.2.1 Input

Seien $A = (a_{ij})$, $B = (b_{ij})$, $C = (c_{ij}) \in \mathbb{R}_+^{n \times m}$ und $K^F = \left(k_{ij}^F\right)$, $K^R = \left(k_{ij}^R\right) \in \mathbb{N}_+^{n \times m}$ Matrizen sowie $P_{cap} = (p_i^*) \in \mathbb{N}_+^n$ und $Q_{cap} = \left(q_j^*\right) \in \mathbb{N}_+^m$ Vektoren. Diese Matrizen und Vektoren stellen den Input für den erweiterten HNS-Algorithmus dar. Sie beinhalten die Informationen wie in 5.1 beschrieben.

5.2.2 Zulässiges Outcome

Das Outcome wird definiert als 6-Tupel $\left(F, R, U^F, U^R, V^F, V^R\right)$. Dabei sind $F := \left(\sigma_{ij}^F\right)$, $R := \left(\sigma_{ij}^R\right) \in \mathbb{N}_+^{n \times m}$ und $U^F := \left(u_{ij}^F\right)$, $U^R := \left(u_{ij}^R\right)$, $V^F := \left(v_{ij}^F\right)$, $V^R := \left(v_{ij}^R\right) \in \mathbb{R}_+^{n \times m}$.

In den Matrizen F und R werden dabei die Kantenkapazitäten gespeichert. Ist zum Beispiel in einem Outcome $\sigma_{ij}^R > 0$, dann besteht in diesem Outcome eine rigide Kante zwischen $i \in P$ und $j \in Q$, die σ_{ij}^R Arbeitszeiteinheiten zwischen i und j matcht.

Ein Outcome heißt zulässig, wenn alle folgenden Bedingungen erfüllt sind:

$$\sum_{j=1}^{m} \sigma_{ij}^R + \sum_{j=1}^{m} \sigma_{ij}^F \leq p_i^* \ (\forall 1 \leq i \leq n) \tag{5.1}$$

$$\sum_{i=1}^{n} \sigma_{ij}^R + \sum_{i=1}^{n} \sigma_{ij}^F \leq q_j^* \ (\forall 1 \leq j \leq m) \tag{5.2}$$

$$\sigma_{ij}^F > 0 \Rightarrow u_{ij}^F + v_{ij}^F = c_{ij} \ (\forall 1 \leq i \leq n, \forall 1 \leq j \leq m) \tag{5.3}$$

$$\sigma_{ij}^R > 0 \Rightarrow u_{ij}^R = a_{ij} \text{ und } v_{ij}^R = b_{ij} \ (\forall 1 \leq i \leq n, \forall 1 \leq j \leq m) \tag{5.4}$$

$$\sigma_{ij}^F \leq k_{ij}^F \ (\forall 1 \leq i \leq n, \forall 1 \leq j \leq m) \tag{5.5}$$

$$\sigma_{ij}^R \leq k_{ij}^R \ (\forall 1 \leq i \leq n, \forall 1 \leq j \leq m) \tag{5.6}$$

5.2.3 Stabiles Outcome

Ausgangspunkt für die Definition eines (paarweise) stabilen Outcomes für dieses Modell ist die Definition eines paarweise stabilen Matchings nach Sotomayor (4.2.1). Bei dieser Definition muss nun allerdings noch berücksichtigt werden, dass auch Kapazitätsrestriktionen auf den einzelnen Kanten bestehen. Diese können dazu führen, dass die Einrichtung oder der Ausbau einer Partnerschaft

für beide Agenten vorteilhaft wäre, aber wegen dieser Begrenzungen einfach nicht zulässig ist.

Wir definieren zur weiteren Formulierung:

$$u^R_{\min,j}(i) := \begin{cases} u^R_{ij} \text{ falls } (i,j)^R \text{ ist die einzige zu } i \text{ inzidente Kante} \\ \min\left(\left(u^F_{ij'} | j' \in Q, \sigma^F_{ij'} > 0\right), \left(u^R_{ij'} | j' \in Q, \sigma^R_{ij'} > 0, j' \neq j\right)\right) \quad \textit{sonst} \end{cases}$$

$$u^F_{\min,j}(i) := \begin{cases} u^F_{ij} \text{ falls } (i,j)^F \text{ ist die einzige zu } i \text{ inzidente Kante} \\ \min\left(\left(u^R_{ij'} | j' \in Q, \sigma^R_{ij'} > 0\right), \left(u^F_{ij'} | j' \in Q, \sigma^F_{ij'} > 0, j' \neq j\right)\right) \quad \textit{sonst} \end{cases}$$

$$v^R_{\min,i}(j) := \begin{cases} v^R_{ij} \text{ falls } (i,j)^R \text{ ist die einzige zu } j \text{ inzidente Kante} \\ \min\left(\left(v^F_{i'j} | i' \in P, \sigma^F_{i'j} > 0\right), \left(v^R_{i'j} | i' \in P, \sigma^R_{i'j} > 0, i' \neq i\right)\right) \quad \textit{sonst} \end{cases}$$

$$v^F_{\min,i}(j) := \begin{cases} v^F_{ij} \text{ falls } (i,j)^F \text{ ist die einzige zu } j \text{ inzidente Kante} \\ \min\left(\left(v^R_{i'j} | i' \in P, \sigma^R_{i'j} > 0\right), \left(v^F_{i'j} | i' \in P, \sigma^F_{i'j} > 0, i' \neq i\right)\right) \quad \textit{sonst} \end{cases}$$

In diesen Definitionen benennt zum Beispiel $u^R_{\min,j}(i)$ das Minimum aller Profite, die ein Knoten i aus den bestehenden inzidenten Kanten außer der Kante $(i,j)^R$ zieht - sofern i noch zu anderen Kanten inzident ist. Völlig analog kann man die anderen Definitionen interpretieren. Die Fallunterscheidung wird notwendig, damit auch der Fall behandelt wird, dass ein Agent lediglich zu einer einzigen Kante inzident ist.

Die vier eben definierten Ausdrücke benötigen wir für die Stabilitätsuntersuchung. Um festzustellen, dass ein Matching paarweise stabil ist, reicht es auszuschließen, dass zwei beliebige Agenten $i \in P$ und $j \in Q$ auch nur eine einzige Kapazität von ihrer jeweils am wenigsten profitablen anderweitigen Partnerschaft in die Kante $(i,j)^R$ beziehungsweise $(i,j)^F$ zu geben bereit sind. Wenn das ausgeschlossen ist, dann folgt daraus, dass i und j auch nicht bereit sind, mehr Kapazitäten zu verschieben oder profitablere Kanten zu streichen. Alle diese Überlegungen müssen dabei nur für solche Agenten i und j durchgeführt werden, die überhaupt anderweitige Partnerschaften besitzen[1] und bei denen nicht $(i,j)^R$ und $(i,j)^F$ bereits voll ausgelastet sind.

Ein Outcome ist mit dem oben Gesagten somit paarweise stabil, wenn für beliebige $i \in P$ mit $p^*_i > 0$ und $j \in Q$ mit $q^*_j > 0$ die folgenden Bedingungen erfüllt sind:

$$\sum_{j'=1}^{m} \sigma^R_{ij'} + \sum_{j'=1}^{m} \sigma^F_{ij'} < p^*_i \wedge \sum_{i'=1}^{n} \sigma^R_{i'j} + \sum_{i'=1}^{n} \sigma^F_{i'j} < q^*_j \Rightarrow \sigma^R_{ij} = k^R_{ij} \wedge \sigma^F_{ij} = k^F_{ij} \quad (5.7)$$

$$\sigma^R_{ij} < k^R_{ij} \Rightarrow a_{ij} \leq u^R_{\min,j}(i) \text{ oder } b_{ij} \leq v^R_{\min,i}(j) \quad (5.8)$$

$$\sigma^F_{ij} < k^F_{ij} \Rightarrow c_{ij} \leq u^F_{\min,j}(i) + v^F_{\min,i}(j) \quad (5.9)$$

Durch diese Bedingungen haben wir die Stabilitätsbedingungen erfasst. Wenn ein Agent $x \in P \cup Q$ noch freie Kapazitäten besitzt, dann gibt es keine zu x

[1] Dabei muss in dem Fall, dass gerade die rigide Kante $(i,j)^R$ betrachtet wird, auch die flexible Kante $(i,j)^F$ als „anderweitige Partnerschaft“ angesehen werden. Denn auch eine Kapazitätsverschiebung zwischen flexibler und rigider Kante muss in einem paarweise stabilen Matching ausgeschlossen werden. Das hier Angemerkte gilt selbstverständlich auch, wenn hier die Begriffe rigide und flexibel vertauscht werden.

inzidenten Kanten, die noch freie Kapazitäten besitzen und zu einem Agenten führen, der selbst noch freie Kapazitäten besitzt. Wenn eine Kante nicht voll ausgelastet ist und beide Endknoten besitzen noch Kapazitäten außerhalb dieser Kante, dann sind nicht beide zur Kante inzidenten Agenten gewillt, auch nur eine Einheit an Kapazitäten von einer anderen bestehenden Kante in diese Kante zu geben.

5.3 Der b-HNS-Algorithmus

5.3.1 Übersicht über den Algorithmus

5.3.1.1 Vorbemerkungen

Für unseren Algorithmus werden wir häufig zur Vereinfachung die Elemente $P = \{1, \ldots, n\}$ als Index der Zeilen und die von $Q = \{1, \ldots, m\}$ als Index der Spalten der oben aufgeführten Matrizen verwenden. Bei den Vektoren P_{cap} beziehungsweise Q_{cap} beschreiben entsprechend die Elemente von P bzw. Q den Index der Zeilen.

Die Inputdaten werden wir mit einem vollständigen bipartiten Graphen $G = (P \dot{\cup} Q, E)$ gleichsetzen mit $|P| = n$ und $|Q| = m$. In einem zulässigen Matching dürfen nur dann zwischen zwei Knoten $i \in P$ und $j \in Q$ zwei Kanten bestehen, wenn eine davon flexibel und eine rigide ist. Einer flexiblen Kante ordnen wir die Gewichtsfunktion (c_{ij}) zu, während zu einer rigiden Kante die Gewichtsfunktion (a_{ij}, b_{ij}) gehört. In der „Flexibles-Matching"-Matrix F und der „Rigides-Matching"-Matrix R ist festgelegt, zwischen welchen Agenten mit welcher Kapazität eine flexible bzw. rigide Kante besteht.

P_{cap} bzw. Q_{cap} definieren die Kapazitäten, die jedem einzelnen Knoten zur Verfügung stehen. Wir werden dem bereits im vorhergehenden Kapitel intuitiv verwendeten Begriff der Kapazität eine doppelte Bedeutung geben. Zum einen bezeichnet Kapazität die Anzahl der Arbeitszeiteinheiten, die jedem Knoten zur Verfügung stehen. Außerdem wird mit Kapazität für jede Kante die Anzahl an Arbeitseinheiten bezeichnet, die diese Kante maximal aufnehmen kann. So gilt insbesondere, dass in einem zulässigen Matching für jeden Knoten y aus P und Q die Summe der Kapazitäten der Kanten mit Endknoten y nicht größer als die Kapazität von y sein darf. Aus dem Zusammenhang wird aber jeweils erkenntlich sein, welche Begriffsauslegung gemeint ist.

Für unsere Überlegungen können wir annehmen, dass $\sum\limits_{P_{cap}} p_i^* = \sum\limits_{Q_{cap}} q_j^*$ gilt. Andernfalls können wir im gegebenen Graphen P oder Q mit einem Knoten erweitern, der ausreichende Kapazitäten besitzt, und erweitern die Präferenzmatrizen passend um eine Nullzeile bzw. Nullspalte. Alle Kanten, die von diesem Knoten ausgehen, erhalten die gleiche Kapazität wie dieser neu hinzugefügte Knoten.

Außerdem können wir annehmen, dass für alle $i \in P$ beziehungsweise $j \in Q$ die Ungleichungen

$$\sum_{j \in Q} \min\left(k_{ij}^F + k_{ij}^R; q_j^*\right) \geq p_i^*$$

beziehungsweise

$$\sum_{i \in P} \min\left(k_{ij}^F + k_{ij}^R; p_i^*\right) \geq q_j^*$$

erfüllt sind. Andernfalls ist bereits vor Beginn des Algorithmus klar, dass ein Agent, der die ihn betreffende Ungleichung nicht erfüllt, entsprechend viele insolvente Kapazitäten haben wird. (Zur Definition von „insolventen Kapazitäten“ siehe weiter unten.) Die überzähligen Kapazitäten eines Agenten können gleich mit dem Profit 0 versehen werden und werden aus der Betrachtung gestrichen. Damit ist allerdings nicht gesagt, dass damit jeder Agent in einem stabilen Outcome alle seine verbleibenden Kapazitäten auch vergeben kann.

Bevor nun der b-HNS-Algorithmus vorgestellt wird, soll noch eine Schreibweise eingeführt werden. Ein hochgestelltes R/F wird verwendet, um Aussagen kürzer zu gestalten. Anstelle von „Es muss $\sigma_{ij}^R \leq k_{ij}^R$ und $\sigma_{ij}^F \leq k_{ij}^F$ gelten.“ wird auch „Es muss $\sigma_{ij}^{R/F} \leq k_{ij}^{R/F}$ gelten.“ geschrieben. R/F kann je nach Zusammenhang als „R und F“ oder auch „R beziehungsweise F“ gelesen werden. Es wurde aber darauf geachtet, dass sich die genaue Bedeutung jeweils aus dem Text erschließt.

5.3.1.2 Ablaufschema des b-HNS-Algorithmus

In weiten Teilen wird der b-HNS-Algorithmus analog zum aus [13] bekannten HNS-Algorithmus verlaufen.

Firmen werden Angebote unterbreiten, Arbeiter werden die ihnen vorliegenden Angebote überprüfen und gegebenenfalls auch ablehnen. Für jeden im Algorithmus gerade betrachteten Agenten soll im entsprechenden Schritt der gerade maximal mögliche Profit angestrebt werden.

Dies sei zuerst am Beispiel einer Firma i erläutert. Der Gesamtprofit S_i von i sei

$$S_i = \sum_{j=1}^{m} \sigma_{ij}^F u_{ij}^F + \sum_{j=1}^{m} \sigma_{ij}^R u_{ij}^R \tag{5.10}$$

Dabei müssen die Beschränkungen (5.1), (5.5) und (5.6) eingehalten werden. Unter der Beachtung dieser Beschränkungen wird nun der Reihe nach für alle i S_i in (5.10) maximiert. Dabei ist es insbesondere möglich, dass die Summe der Kapazitäten aller Kanten, die zu einem Arbeiter j gehören, größer als q_j^* ist. Die Einhaltung dieser Nebenbedingung wird an einer anderen Stelle des Algorithmus angegangen.

Für einen Arbeiter j soll ebenfalls der Profit maximiert werden. Im Unterschied zu Firmen unterbreiten Arbeiter keine Angebote. Für j werden aus den vorliegenden Angeboten diejenigen heraus gesucht, die für j den maximalen Profit ergeben. Dabei können dann gegebenenfalls auch Profite aus flexiblen Angeboten erhöht werden.

Wie in klassischen Modellen mit flexiblen Kanten werden auch hier Arbeitern flexible Kanten nur in der Form angeboten, dass Firmen einen möglichst hohen Profit erhalten. Allerdings können Profite aus flexiblen Kanten, wie erwähnt, im Laufe des Algorithmus zu Gunsten der Arbeiter verschoben werden, nie jedoch in die andere Richtung. Dies ist der Grund, weshalb bei jeweils gleichem Profit für die Kapazitätsdeckung von Firmen rigide Kanten bevorzugt werden, für die Kapazitätsdeckung von Arbeitern allerdings flexible Kanten.

Im ersten Schritt macht nun jede Firma i denjenigen Arbeitern Angebote, die den erwarteten Profit der Firma i maximieren. Dies geschieht im Algorithmus PlaceProposals. Aufgrund der Kapazitätsbeschränkungen bei den Kanten kann es durchaus sein, dass eine Firma mehreren Arbeitern Angebote macht. Ein Angebot wird nur unterbreitet, wenn der erwartete Profit strikt positiv ist. Kann die Firma wegen dieser Regelung in jedem stabilen Outcome nicht zu allen ihren Kapazitäten Angebote unterbreiten, so nennen wir die Firma ganz oder teilweise insolvent. Andernfalls heißt sie solvent. Insolvente Kapazitäten bleiben im Algorithmus unberücksichtigt und werden am Ende des Algorithmus (fast) willkürlich mit Arbeitern mit freien Kapazitäten gematcht.

Hat jede Firma ihre Angebote unterbreitet, wählen die Arbeiter nun die Angebote aus, die ihre Kapazitäten mit maximalem Profit ausfüllen. Dies wird im später noch genauer erläuterten Unteralgorithmus CheckProposals erfolgen. Dabei werden bei gleichem Profit flexible Angebote rigiden Angeboten vorgezogen. Bei den gegebenenfalls vorhandenen *überschüssigen Angeboten*, das heißt Angeboten, die ein Arbeiter wenig bevorzugt und deren Streichung oder Reduzierung nicht dazu führt, dass ein Arbeiter wieder freie Kapazitäten hat, werden alle Kapazitäten aus rigiden Angeboten abgelehnt.

Dieser Vorgang aus Angebotsunterbreitung von Seiten der Firmen und Angebotsabwägung von Seiten der Arbeiter wird solange fortgesetzt, bis jede Firma einen Abnehmer für ihre solventen Kapazitäten gefunden hat und kein Arbeiter überschüssige rigide Kanten besitzt.

Für den Fall, dass noch Arbeiter existieren, bei denen die angebotenen Kapazitäten die dem Arbeiter zur Verfügung stehenden Kapazitäten übersteigen, fahren wir fort. Wir suchen Netzwerke in einem speziell zu konstruierenden Digraphen, mit denen wir über Alternierungen unter anderem Kapazitäten von einem Arbeiter mit überschüssigen Kapazitäten zu einem Arbeiter mit freien Arbeitszeiteinheiten verschieben können.

Falls dann noch immer Arbeiter mit zu vielen angebotenen Kapazitäten existieren, führen wir eine modifizierte Version des HungarianUpdate durch. Dadurch entstehen im oben konstruierten Digraphen möglicherweise neue Netzwerke zum Kapazitätsausgleich. Wir können nun den Algorithmus erneut durchlaufen lassen, bis alle solventen Kapazitäten der Firmen so untergebracht wurden, dass bei keinem Arbeiter überschüssige Kapazitäten bestehen.

5.3.2 Detaillierte Strukturierung des b-HNS-Algorithmus

Das eben dargelegte allgemeine Vorgehen im Algorithmus soll nun näher erläutert werden.

Es sei noch eine Hilfsmatrix $N := (n_{ij}) \in \{0,1\}^{n \times m}$ definiert und mit $N = 0$ initialisiert. Diese Matrix wird notwendig, um bei der folgenden Situation zu definieren, wie der Algorithmus reagieren soll, und um damit eine eventuelle Endlosschleife zu verhindern:

Einer Firma i stehen nach Durchführung von PlaceProposals noch Kapazitäten zur Verfügung. Außerdem besitzt sie eine rigide Kante $(i,j)^R$, deren Kapazität noch ausgebaut werden kann, das heißt es gilt $\sigma_{ij}^R < k_{ij}^R$. Es muss definiert sein, ob in diesem Fall i weitere Kapazitäten in die Kante $(i,j)^R$ geben soll. Denn es können zwei Vorgänge zu dieser Kante mit nicht maximaler Kapazität führen:

1. i hatte das Angebot ursprünglich nur mit begrenzten Kapazitäten unterbreitet, weil andere Kanten profitabler waren. Nachdem im Laufe des Algorithmus andere Angebote abgelehnt wurden, wird ein Ausbau der Kante $(i,j)^R$ für i wieder interessant.

2. Die Kapazität von $(i,j)^R$ wurde von j in CheckProposals reduziert.

Dabei können beide Fälle zusammen auftreten. Während im Fall 1 eine Kapazitätserweiterung des bestehenden Angebots $(i,j)^R$ auch für j sinnvoll sein kann, ist klar, dass im Fall 2 jede neue Antragstellung durch i von j wieder abgelehnt werden würde. Damit der Algorithmus terminiert, muss also sichergestellt sein, dass i kein Angebot an j unterbreitet wird, wenn der Fall 2 vorliegt. Dies geschieht durch die oben definierte Matrix N. i stellt höchstens an die Firmen rigide Angebote, für die $n_{ij} = 0$ ist.

Für flexible Kanten ist eine entsprechende Hilfsmatrix im Übrigen nicht notwendig. Flexible Kanten werden im Algorithmus nicht so einfach gestrichen wie rigide Kanten. Entweder werden sie durch neue Kanten mit gleicher Kapazität wie die gestrichenen Kapazitäten ersetzt oder Kapazitäten einer Firma werden im gleichen Umfang als insolvent markiert und werden somit nicht mehr neu verteilt.

Vor Beginn des eigentlichen Algorithmus initialisieren wir noch

$$V^F = V^R = 0.$$

Alle $j \in Q$ besitzen also zu Beginn des Algorithmus den Profit 0. Hierbei handelt es sich ganz offensichtlich um ein zulässiges Outcome.

Im Verlauf des Algorithmus werden die Einträge im Vektor P_{cap} variieren. Die Einträge sollen jeweils anzeigen, wie viele Kapazitäten einer Firma noch zur Verfügung stehen. Da die ursprünglichen Einträge in diesem Vektor weiterhin relevant sind, setzen wir $P_{cap}^{Anf} := \left(p_i^{Anf}\right)_n := P_{cap}$.

5.3.2.1 Erster Schritt: PlaceProposals

Zuerst wird der Algorithmus PlaceProposals durchgeführt. Dabei bietet jede Firma aus P im ersten Schritt ihre gesamten Kapazitäten bestimmten Arbeitern an. Hier werden nur solvente Firmen beziehungsweise deren solvente Kapazitäten berücksichtigt. Insolvente Kapazitäten werden zum Ende des Algorithmus willkürlich, aber unter Beachtung der Kapazitätsbegrenzungen der zugehörigen Endknoten, mit Arbeitern mit Restkapazität mittels einer rigiden Kante verbunden oder, falls dies nicht möglich ist, mittels flexibler Kanten. Dennoch können Kapazitäten verbleiben, die nicht gematcht sind.

Bei PlaceProposals gehen wir für eine Firma i im Detail wie folgt vor:

Wir sortieren alle $2m$ Kanten zu denen i inzident sein kann mit einer $\geq$-Ordnung. Für je zwei Kanten $(i,j_1)^{X_1}$ und $(i,j_2)^{X_2}$ mit $X_1, X_2 \in \{R,F\}$ gilt dabei

$$\begin{aligned} u_{ij_1}^{X_1} > u_{ij_2}^{X_2} &\Rightarrow (i,j_1)^{X_1} > (i,j_2)^{X_2} \\ u_{ij_1}^{X_1} = u_{ij_2}^{X_2} \text{ und } X_1 = R, X_2 = F &\Rightarrow (i,j_1)^{X_1} > (i,j_2)^{X_2} \\ u_{ij_1}^{X_1} = u_{ij_2}^{X_2} \text{ und } X_1 = X_2 &\Rightarrow (i,j_1)^{X_1} = (i,j_2)^{X_2} \end{aligned}$$

Mit jedem Angebot einer Firma i wird auch der Eintrag p_i^* um die vergebenen Kapazitäten reduziert. Wird zu einem späteren Zeitpunkt ein solches Angebot abgelehnt, wird p_i^* wieder entsprechend erhöht. Somit zeigt der Eintrag bei p_i^* immer an, wie viele Kapazitäten der Firma i noch zu vergeben sind.

Anschließend wird die Menge D_i definiert. In diese Menge werden all die Kanten aufgenommen, in die i seine gesamten solventen Kapazitäten mit maximalem Profit geben kann. Damit bleiben Kanten mit einem Profit von 0 für i unberücksichtigt. Es werden also die Kanten in D_i aufgenommen, die mit der obigen Ordnung am größten sind. Gilt für zwei Kanten $(i, j_1)^{X_1}$ und $(i, j_2)^{X_2}$, dass sie den gleichen Profit für i erbringen, und ist eine der beiden Kanten in D_i enthalten, dann wird auch die anderen Kante in D_i aufgenommen. Zum Abschluss werden mit den Kanten aus D_i die Kapazitäten von i profitmaximierend vergeben. Da nur strikt positive Profite für i berücksichtigt werden, kann es sein, dass nicht alle Kapazitäten von i in Angeboten untergebracht werden können.

Da wir diesen Wert im späteren Verlauf des Algorithmus noch benötigen, definieren wir noch d_i als den maximalen Profit aller zu i inzidenten Kanten, die nicht in D_i enthalten sind. Falls keine solche Kante existiert, setze $d_i = 0$.

Dann werden die Knoten $j \in Q$ untersucht. Als erstes setzen wir $v_{ij}^R = b_{ij}$, falls das Angebot von i an j rigide ist.

Wir definieren für jedes $j \in Q$ drei Mengen, die jeweils eine strenge Ordnungsrelation[2] erhalten. Die Elemente dieser Menge sind Tupel $\left(i, \sigma_{ij}^X\right)$ mit $X \in \{R, F\}$. Dabei ist $i \in P$ und $\sigma_{ij}^{R/F}$ die Kantenkapazität von $(i, j)^{R/F}$. Wir definieren die Menge Z_j wie folgt:

$$\left(i, \sigma_{ij}^{R/F}\right) \in Z_j \Leftrightarrow \sigma_{ij}^{R/F} > 0. \tag{5.11}$$

Z_j entsteht also aus allen Kanten mit Endknoten j. Für $i_1 \neq i_2$ gelte $\left(i_1, \sigma_{i_1 j}^{X_1}\right) > \left(i_2, \sigma_{i_2 j}^{X_2}\right)$ mit $X_1, X_2 \in \{R, F\}$, wenn

$$v_{i_1 j}^{X_1} > v_{i_2 j}^{X_2} \tag{5.12}$$

oder

$$v_{i_1 j}^{X_1} = v_{i_2 j}^{X_2} \text{ und } X_1 = F \text{ und } X_2 = R \tag{5.13}$$

gilt.

Für die beiden Tupel $\left(i, \sigma_{ij}^R\right)$ und $\left(i, \sigma_{ij}^F\right)$ gilt, sofern sie beide in Z_j enthalten sind:

$$\left(i, \sigma_{ij}^R\right) > \left(i, \sigma_{ij}^F\right), \text{ falls } v_{ij}^R > v_{ij}^F \tag{5.14}$$

und

$$\left(i, \sigma_{ij}^F\right) > \left(i, \sigma_{ij}^R\right), \text{ falls } v_{ij}^R \leq v_{ij}^F. \tag{5.15}$$

Allen Tupeln aus Z_j, die nicht über die obigen Fälle definiert sind, wird willkürlich eine $>$-Relation zugeordnet. Die einzige Bedingung, die dabei gestellt wird, ist, dass eine solche Zuordnung transitiv sein muss. Das heißt aus

$$\left(i_1, \sigma_{i_1 j}^{X_1}\right) > \left(i_2, \sigma_{i_2 j}^{X_2}\right) \text{ und } \left(i_2, \sigma_{i_2 j}^{X_2}\right) > \left(i_3, \sigma_{i_3 j}^{X_3}\right)$$

[2] zur Definition einer strengen Ordnungsrelation siehe zum Beispiel [20]

muss für alle Elemente von Z_k

$$\left(i_1, \sigma_{i_1 j}^{X_1}\right) > \left(i_3, \sigma_{i_3 j}^{X_3}\right)$$

folgen.

Die Menge Z_j^* entsteht aus Z_j wie folgt:

Falls $\sum_{i=1}^{n} \sigma_{ij}^R + \sum_{i=1}^{n} \sigma_{ij}^F \leq q_j^*$, ist $Z_j^* = Z_j$.

Wähle sonst aus Z_j die gemäß der obigen Ordnungsrelation größten Elemente $\left(i, \sigma_{ij}^X\right)$, $X \in \{R, F\}$ aus und füge sie einzeln Z_j^* hinzu, bis q_j^* gerade erreicht oder überschritten wird. Nun werden die in den Tupeln von Z_j^* enthaltenen Kapazitäten modifiziert. Ersetze $\sigma_{ij}^{R/F}$ durch $\tau_{ij}^{R/F}$. Dabei ist $\tau_{ij}^{R/F} := \sigma_{ij}^{R/F}$ für alle bis auf das zuletzt hinzugefügte Tupel. Dieses Tupel zeichnen wir besonders aus mit $\left(\widetilde{i}, \sigma_{\widetilde{i}j}^X\right)$, wobei wieder $X \in \{R, F\}$ gilt. Für dieses Tupel setze $\tau_{\widetilde{i}j}^X$ so, dass die Gleichung

$$\sum_{\left(i, \tau_{ij}^X\right) \in Z_j^*} \tau_{ij}^X = q_j^*$$

erfüllt ist. Sofern in $\left(\widetilde{i}, j\right)^X$ $X = R$ gilt, das heißt es handelt sich um eine rigide Kante, setzen wir anschließend $\sigma_{\widetilde{i}j}^X = \tau_{\widetilde{i}j}^X$ und $n_{\widetilde{i}j} = 1$. Wir erreichen dadurch, dass eine eventuell „überstehende“ rigide Kante aus Sicht von j „passend“ gekürzt wird. $p_{\widetilde{i}}^*$ wird in diesem Fall entsprechend erhöht.

Z_j^* beinhaltet somit eine Auswahl der Kanten, mit denen die Kapazitäten von j aus den vorliegenden Angeboten mit maximalem Profit gefüllt werden können.

Die Menge Z_j' entsteht aus Z_j wie folgt:

Falls $\sum_{i=1}^{n} \sigma_{ij}^R + \sum_{i=1}^{n} \sigma_{ij}^F < q_j^*$, ist $Z_j' = \emptyset$.

Falls $\sum_{i=1}^{n} \sigma_{ij}^R + \sum_{i=1}^{n} \sigma_{ij}^F = q_j^*$, ist $Z_j' = \left\{\left(\widetilde{i}, \sigma_{\widetilde{i}j}^X\right)\right\}$.

Falls $\sum_{i=1}^{n} \sigma_{ij}^R + \sum_{i=1}^{n} \sigma_{ij}^F > q_j^*$, nimm in Z_j' all die Tupel auf, die bei der Konstruktion von Z_j^* nicht berücksichtigt wurden. Füge außerdem das Tupel $\left(\widetilde{i}, \sigma_{\widetilde{i}j}^X\right)$ hinzu.

Z_j' vereinfacht das Problem deutlich, wie wir sehen werden. Denn über Z_j' kann das Problem auf den bekannten HNS-Algorithmus zurückgeführt werden.

Der Unteralgorithmus CheckProposals

Dieser Unteralgorithmus wird nun von allen Knoten aus Q durchlaufen, bei denen

$$\sum_{i=1}^{n} \sigma_{ij}^R + \sum_{i=1}^{n} \sigma_{ij}^F > q_j^*$$

gilt. Wir führen dafür die bereits im vorangehenden Text intuitiv verwendete Sprechweise ein: *j hat überschüssige Kapazitäten.*

Suche alle Tupel $\left(i', \sigma^R_{i'j}\right) \in Z'_j$ mit $\left(i', \sigma^R_{i'j}\right) \neq \left(\widetilde{i}, \sigma^X_{\widetilde{i}j}\right)$. Setze für solche Kanten $\sigma^R_{i'j} = 0$, $n_{i'j} = 1$ und entferne das zugehörige Tupel aus Z_j und Z'_j. Der Wert $p^*_{i'}$ wird um die Kapazität der gestrichenen Kante erhöht.

Falls in $\left(\widetilde{i}, \sigma^X_{\widetilde{i}j}\right)$ $X = R$ gilt, verfahre wie folgt: Setze für alle $i \in P$ mit $\left(i, \tau^F_{ij}\right) \notin Z^*_j$ $v^F_{ij} = \min\left(v^R_{\widetilde{i}j}, c_{ij}\right)$. Setze außerdem für alle $i \in P$ mit $\left(i, \tau^R_{ij}\right) \notin Z^*_j$ $v^R_{ij} = v^R_{\widetilde{i}j}$. Reduziere u^F_{ij} aller durch diese Maßnahmen betroffenen $i \in P$ mit einer bestehenden Kante zu j um den gleichen Betrag, um den v^F_{ij} angehoben wurde. Es sei angemerkt, dass hier keine Werte aus V^R und U^R geändert werden, die zu einer bestehenden rigiden Kante mit Endknoten j gehören.

Lösche diejenigen Kanten, das heißt setze $\sigma^F_{i''j} = 0$, die eine Firma i'' aufgrund des geänderten $u^F_{i''j}$ nicht mehr zur Abdeckung ihrer Kapazitäten bevorzugt. Dies ist dann der Fall, wenn $u^F_{i''j} \leq d_i$. Insbesondere gehören alle Kanten dazu, die i'' den Profit 0 bringen. Erhöhe $p^*_{i''}$ entsprechend den so gestrichenen Kapazitäten.

Konstruiere abschließend die Mengen Z_j, Z^*_j und Z'_j neu. Gehe dabei wie oben beschrieben vor. Beachte, dass sich in der Regel $\left(\widetilde{i}, \sigma^X_{\widetilde{i}j}\right)$ ändert.

Nach einmaligem Durchführen dieser Schritte gilt bereits, dass $X = F$ in $\left(\widetilde{i}, \sigma^X_{\widetilde{i}j}\right)$ oder es ist $\sum_{i=1}^{n} \sigma^R_{ij} + \sum_{i=1}^{n} \sigma^F_{ij} \leq q^*_j$ für alle $j \in Q$. Der Unteralgorithmus CheckProposals ist somit abgeschlossen.

Durch CheckProposals wird der Profit jedes dort betrachteten Arbeiters j unter Ausnutzung der vorliegenden Angebote maximiert. Durch die Anhebung bestimmter, bislang nicht realisierter flexibler Profite können sukzessive flexible Angebote anstelle von rigiden Angeboten zur Deckung der Kapazitäten herangezogen werden. Wie bereits oben beschrieben, werden zur Profitoptimierung von $j \in Q$ flexible Kanten gegenüber rigiden bevorzugt. Eine einmal erfolgte Deckung des Kapazitätsbedarfs von j geht durch CheckProposals nicht wieder verloren.

Sind wir für alle $j \in Q$ CheckProposals durchgegangen, definieren wir Funktionen $u^F_{\sigma,v}$ und $u^R_{\sigma,v}$:

$$u^F_{\sigma,v}(i,j) := c_{ij} - v^F_{ij} \tag{5.16}$$

$$u^R_{\sigma,v}(i,j) := \begin{cases} a_{ij} & \text{falls } v^R_{ij} < b_{ij} \text{ oder } \sigma^R_{ij} > 0 \\ 0 & \text{sonst} \end{cases} \tag{5.17}$$

Die Funktion $u^R_{\sigma,v}$ stellt zusammen mit der Matrix N sicher, dass i ein abgelehntes (rigides) Angebot nicht wieder stellt. Anhand der Funktionen $u^{R/F}_{\sigma,v}$, die zum Ausdruck bringen, welchen Profit i je Arbeitszeiteinheit von einem Angebot an j erwarten kann, machen nun alle i, die noch freie, solvente Kapazitäten besitzen, über PlaceProposals erneut Angebote an Arbeiter j. Dabei werden bestehende rigide Angebote $(i,j)^R$ nur dann ausgebaut, falls $\sigma^R_{ij} < k^R_{ij}$ und $n_{ij} = 0$.

Dieser Prozess wird fortgeführt, bis keine Firma mehr freie solvente Kapazitäten und jeder Arbeiter j höchstens q^*_j Kapazitäten von rigiden Angeboten besitzt.

5.3.2.2 Zweiter Schritt: Alternierungen im Digraphen

Konstruktion des Digraphen $G^{\sigma,v}$

Für den Fall, dass $\sum_{i=1}^{n} \sigma_{ij}^{R} + \sum_{i=1}^{n} \sigma_{ij}^{F} \leq q_j^*$ noch nicht für alle $j \in Q$ erfüllt ist, gehen wir weiter und versuchen, dies durch Alternierungen zu erreichen. Dazu konstruieren wir einen bipartiten Digraphen $G^{\sigma,v}$ mit Knotenmenge $P \dot\cup Q$. Die Kanten, die sich aus den Elementen der Z_j' aller $j \in Q$ ergeben, entsprechen in diesem Digraphen Pfeilen von Q nach P („rückwärts gerichtete Pfeile"). Die Pfeile von P nach Q („vorwärts gerichtete Pfeile") sind genau die Elemente der Menge

$$D_{\sigma,v}^{i} := \left\{ (i,j)^R \,|a_{ij} = u_{\sigma,v}^{R}(i,j) = \max_{k=1}^{m} \max_{X \in \{R,F\}} u_{\sigma,v}^{X}(i,k) > 0, \sigma_{ij}^{R} < k_{ij}^{R}, n_{ij} = 0 \right\}$$
$$\cup \left\{ (i,j)^F \,|c_{ij} - v_{ij} = u_{\sigma,v}^{F}(i,j) = \max_{k=1}^{m} \max_{X \in \{R,F\}} u_{\sigma,v}^{X}(i,k) > 0, \sigma_{ij}^{F} < k_{ij}^{F} \right\}$$

Dabei kann es passieren, dass im Graphen $G^{\sigma,v}$ bis zu vier Pfeile aus nur zwei Knoten $i \in P$ und $j \in Q$ resultieren. Wenn zwischen i und j rigide Pfeile in beide Richtungen entstehen, streichen wir den Pfeil $(i,j)^R$ und setzen $n_{ij} = 1$. Dieses Vorgehen macht Sinn, denn andernfalls hieße das, dass die rigide Kante $(i,j)^R$ weiter ausgebaut werden könnte. Im Graphen $G^{\sigma,v}$ ist nur dann der Pfeil $(j,i)^R$ enthalten, wenn in Z_j $\left(i, \sigma_{ij}^{R}\right) = \left(\widetilde{i}, \sigma_{\widetilde{i}j}^{X}\right)$ gilt. Dann hat aber j bereits ausreichend Kapazitäten und die so hinzugekommenen Kapazitäten würden in CheckProposals wieder gelöscht werden. Wenn in $G^{\sigma,v}$ ein rigider Pfeil $(i,j)^R$ und ein flexibler Pfeil $(j,i)^F$ vorliegen, werden wir sofort aktiv. Wir verschieben $\max\left\{\sigma_{ij}^{R}, k_{ij}^{F} - \sigma_{ij}^{F}\right\}$ von der rigiden Kante $(i,j)^R$ in die flexible Kante $(i,j)^F$ und aktualisieren die betroffenen Einträge zu den Kapazitäten. Dadurch verschwindet mindestens einer der beiden Pfeile aus $G^{\sigma,v}$.

Im Digraphen $G^{\sigma,v}$ beschreibt nun $\sigma_{ij}^{R/F}$ die Kapazität eines rückwärts gerichteten Pfeils. Die Kapazität eines vorwärts gerichteten Pfeils $(i,j)^R$ ist

$$cap_R(i,j) = k_{ij}^{R} - \sigma_{ij}^{R}. \tag{5.18}$$

Entsprechend gilt für den vorwärts gerichteten Pfeil $(i,j)^F$, dass dessen Kapazität

$$cap_F(i,j) = k_{ij}^{F} - \sigma_{ij}^{F} \tag{5.19}$$

ist.

Ermitteln von Netzwerken

Sei nun $j_0 \in Q$ ein Knoten mit $\sum_{i=1}^{n} \sigma_{ij_0}^{R} + \sum_{i=1}^{n} \sigma_{ij_0}^{F} > q_{j_0}^*$. Die folgenden Schritte werden solange wiederholt, bis kein solcher Knoten mehr existiert oder bis in $G^{\sigma,v}$ kein Netzwerk $\mathcal{N}$ mit Quelle j_0 existiert, das eines der folgenden Eigenschaften besitzt:

1. $\mathcal{N}$ besitzt ausschließlich flexible Pfeile und endet in der Senke $j_1 \in Q$. In $G^{\sigma,v}$ existiert ein rigider Pfeil mit Anfangsknoten j_1.

2. $\mathcal{N}$ besitzt ausschließlich flexible Pfeile und endet in der Senke $i \in P$, zu dessen Traumpartnern ein $j_1 \in Q$ gehört und der Pfeil in $G^{\sigma,v}$ zwischen i und j_1 ist rigide.

3. $\mathcal{N}$ besitzt ausschließlich flexible Pfeile und endet in der Senke $i \in P$ und alle Kanten, die zu den Pfeilen gehören, über die i in $\mathcal{N}$ erreicht werden kann, bringen i den Profit 0 und i besitzt keine weiteren Vorwärtspfeile in $G^{\sigma,v}$.

4. $\mathcal{N}$ besitzt ausschließlich flexible Pfeile und endet in der Senke $j_1 \in Q$ für die $\sum_{i=1}^{n} \sigma_{ij_1}^{R} + \sum_{i=1}^{n} \sigma_{ij_1}^{F} < q_{j_1}^{*}$ gilt.

Definiere für alle Netzwerke $\sigma_{\mathcal{N}}$ als die Kapazität des Netzwerkes $\mathcal{N}$, das heißt den maximalen Fluss, der in dem Netzwerk von der Quelle j_0 zur jeweils betrachteten Senke konstruiert werden kann. Es ist klar, dass die Quelle j_0 genau $\left(\sum_{i=1}^{n} \sigma_{ij_0}^{R} + \sum_{i=1}^{n} \sigma_{ij_0}^{F}\right) - q_{j_0}^{*}$ Kapazitäten abgeben kann. Der Bedarf der Senke im Netzwerk hängt davon ab, welcher der obigen Fälle vorliegt. Im Fall 1 ist der Bedarf des Knotens $j_1 \in Q$ gerade so hoch wie die Kapazität des rigiden Pfeils mit Anfangsknoten j_1. Im Fall 2 entspricht der Bedarf der Senke gerade der Kapazität der Kante $(i, j_1)^R$. In Fall 3 ist der Bedarf der Senke i so hoch wie die Kapazitäten von i, die noch nicht als insolvent markiert sind, derzeit den Profit 0 haben und für die kein Vorwärtspfeil in $G^{\sigma,v}$ besteht. Im obigen Fall 4 ist $q_{j_1}^{*} - \left(\sum_{i=1}^{n} \sigma_{ij_1}^{R} + \sum_{i=1}^{n} \sigma_{ij_1}^{F}\right)$ die Kapazität der Senke.

Mittels des *Scaling max-flow Algorithmus* [3] (im folgenden mit ScalingMaxFlow bezeichnet) wird unter Berücksichtigung der für $G^{\sigma,v}$ definierten Kantenkapazitäten ein maximaler Fluss von der Quelle j_0 zur jeweils betrachteten Senke ermittelt. Dabei sehen wir den Untergraphen von $\mathcal{N}$, der alle möglichen Pfade von j_0 zum Endknoten $x \in P \cup Q$ beinhaltet, als Netzwerk an.

Ein im Anschluss durchgeführtes CheckProposals führt für den Fall, dass einem Knoten aus Q Kapazitäten zugeführt wurden, zu einer für j_1 optimalen Auswahl aus den vorliegenden Angeboten.

Anschließend werden die beiden Subroutinen DisposeRigid und Alternate durchgeführt. Hierbei passiert folgendes:

DisposeRigid: Nur wenn ein Netzwerk mit Senke $j_1 \in Q$ existiert, in dem alle Pfeile des Netzwerks flexibel sind und j_1 ist in $G^{\sigma,v}$ Anfangsknoten eines rigiden Pfeils $(j_1, i_1)^R$, gehe wie folgt vor:

Reduziere $\sigma_{i_1 j_1}^{R}$ um das Minimum aus den dazugehörenden $\sigma_{\mathcal{N}}$ und $\sigma_{i_1 j_1}^{R}$. Erhöhe $p_{i_1}^{*}$ um den gleichen Betrag und setze $n_{i_1 j_1} = 1$. Hat das Netzwerk $\mathcal{N}$ also ausreichende Kapazität, wird die bestehende rigide Kante zu j_1 gelöscht. Andernfalls wird sie soweit wie möglich reduziert.

Alternate: Alle zum mit ScalingMaxFlow konstruierten Fluss gehörenden Pfeile werden alterniert. Kanten, die zu Pfeilen von Q nach P gehören, werden damit um den ermittelten Wert reduziert und entfallen gegebenenfalls ganz. Kanten, die zu Pfeilen von P nach Q gehören, erhalten analog mehr Kapazitäten.

Schon der Algorithmus PLACEPROPOSALS stellt sicher, dass der Profit $v_{ij}^{R/F}$ für ein festes j bei allen hier in Betracht kommenden i gleich ist. Daher ändert sich der Profit $v_{ij}^{R/F}$ nur dann, wenn j_1 in $G^{\sigma,v}$ ungematcht war. In beiden Fällen wird $v_{ij}^{R/F}$ echt größer. Die Funktionen $u_{\sigma,v}^{R}$ und $u_{\sigma,v}^{F}$ stellen sicher, dass jede Firma i den erwarteten Profit bezüglich der neu entstandenen Kante $(i,j)^{R/F}$ erhält.

Wurden DISPOSERIGID und ALTERNATE je einmal durchgeführt, wird PLACEPROPOSALS erneut ausgeführt. Dies ist notwendig, da unter Umständen rigide Angebote von Firmen ersatzlos gestrichen wurden. Man beachte, dass durch den damit auch ablaufenden Algorithmus CHECKPROPOSALS für alle Firmen Z_j, Z_j^* und Z_j' neu ermittelt werden.

Abschließend wird noch $G^{\sigma,v}$ aktualisiert.

5.3.2.3 Dritter Schritt: Modifiziertes HUNGARIANUPDATE

Überschreitet ein Knoten $j \in Q$ weiterhin seine Kapazitätsrestriktionen, kann aber die überschüssigen Kapazitäten nicht vollständig mit den eben durchgeführten Schritten reduzieren, muss das modifizierte HUNGARIANUPDATE durchgeführt werden.

Betrachte dazu die Komponente in $G^{\sigma,v}$, zu der j gehört. Notwendig können nicht alle Knoten aus Q zu dieser Komponente gehören, andernfalls würden insbesondere auch Arbeiter mit noch freien Kapazitäten dazu gehören und man hätte in $G^{\sigma,v}$ Kapazitäten von j zu einem solchen ungematchten Arbeiter verschieben können.

Wir bezeichnen mit $\overline{P} \subseteq P$ die Knoten aus P, die von j aus erreichbar sind, und mit $\overline{Q} \subset Q$ die Knoten aus Q, die in der zu j gehörenden Komponente liegen.

Nun definieren wir:

$$u_i := \max_{k=1}^{m} \max_{X \in \{R,F\}} u_{\sigma,v}^{X}(i,k) \;\; \forall i \in \overline{P} \tag{5.20}$$

$$\Delta_1 := \min \left\{ u_i - \max_{X \in \{R,F\}} u_{\sigma,v}^{X}(i,j) \,|\, i \in \overline{P}, j \notin \overline{Q} \right\} \tag{5.21}$$

$$\Delta_2 := \min \left\{ u_i - u_{\sigma,v}^{R}(i,j) \,|\, i \in \overline{P}, j \in \overline{Q} \right\} \tag{5.22}$$

$$\Delta_3 := \min \{u_i\} \tag{5.23}$$

Man kann leicht feststellen, dass Δ_1, Δ_2 und Δ_3 jeweils größer als 0 ist. Ein Beweis dieser Aussage ist in Satz 5.5.7 enthalten. Setze

$$\Delta := \min \{\Delta_1, \Delta_2, \Delta_3\} \tag{5.24}$$

und addiere Δ zu allen $v_{ij}^{R/F}$, für die gilt, dass $j \in \overline{Q}$ und $\left(i, \sigma_{ij}^{F}\right) \notin \left(Z_j^* \setminus \left\{\widetilde{i}, \sigma_{\widetilde{i}j}^{X}\right\}\right)$. Wir bemerken, dass wir hier insbesondere keinen Profit einer bestehenden rigiden Kante geändert haben. Über $u_{\sigma,v}^{R/F}$ sinken auch Profite der Firmen aus $\overline{P}$. Durch diese Änderung kommt wenigstens ein neuer vorwärts gerichteter Pfeil in $G^{\sigma,v}$ dazu oder es werden Kapazitäten einer Firma als insolvent markiert.

Nach Abschluss des HUNGARIANUPDATE wird daher der Digraph $G^{\sigma,v}$ aktualisiert. Im geänderten Graphen werden nun wieder Netzwerke für Alternierungen gesucht. Der Algorithmus kehrt also wieder zum Schritt zwei zurück.

Zum Ende des Algorithmus werden schließlich noch die als insolvent markierten Kapazitäten von Firmen nach Möglichkeit in Kanten zu Arbeitern mit offenen Kapazitäten gegeben. Dabei sind einige Vorüberlegungen zu machen. Im Unterschied zum klassischen HNS-Algorithmus können wir hier nicht Arbeiter und Firmen mit offenen Kapazitäten willkürlich matchen. Der Grund liegt insbesondere in den Kantenrestriktionen. Diese führen auch dazu, dass nicht notwendig jeder Agent alle seine Kapazitäten in Kanten geben kann.

In welchen Fällen treten nun freie Kapazitäten nach Durchlaufen aller hier genannten Schritte auf? Sei dazu $i' \in P$ eine Firma, der noch Kapazitäten zur Verfügung stehen. Weil wir in unserem Modell annehmen, dass $\sum\limits_{P_{cap}} p_i^* = \sum\limits_{Q_{cap}} q_j^*$ gilt, muss es auch einen Arbeiter $j' \in Q$ geben, der ebenfalls freie Kapazitäten besitzt.

Sofern für die beiden Kanten zwischen i' und j' $\sigma^R_{i'j'} = k^R_{i'j'}$ und $\sigma^F_{i'j'} = k^F_{i'j'}$ gilt, ist klar, dass die beiden Agenten während des Algorithmus keine Kapazitäten mehr in eine der beiden Kanten geben konnten.

Nehmen wir daher an, mindestens eine der beiden Kanten hat noch keine volle Kantenkapazität. Aus welchem Grund wurden dann die zur Verfügung stehenden Kapazitäten nicht im Laufe des Algorithmus in diese Kante gegeben? Es kann nicht sein, dass im Laufe des Algorithmus Kapazitäten aus dieser Kante gestrichen wurden. Dies passiert nämlich nur in zwei Fällen:

1. Wenn j' bereits volle Kantenkapazitäten hatte. Das widerspricht aber unserer Annahme, dass j' noch freie Kapazitäten besitzt. Denn wie wir beim Beweis des Algorithmus sehen werden, geht eine einmal erreichte volle Kapazitätsauslastung eines Arbeiters während des Algorithmus nicht mehr verloren.

2. Durch Alternierungen, wenn der aus der Kante zwischen i' und j' resultierende (rückwärts gerichtete) Pfeil in einem „passenden“ Netzwerk vorkommt. Damit dies der Fall ist, muss $Z'_{j'} \neq \emptyset$ gelten. Dies ist aber nicht der Fall, denn das hieße, dass j' keine freien Kapazitäten besitzt.

Weil j' noch freie Kapazitäten besitzt, flexible Angebote im Verlaufe des Algorithmus aber höchstens dann einen strikt positiven Profit für einen Arbeiter bringen, wenn dieser keine freien Kapazitäten mehr besitzt, gilt:

Bei einem rigiden Angebot von i' an j' wäre der Profit für i' gleich $a_{i'j'}$, bei einem flexiblen Angebot wäre dieser Profit gleich $c_{i'j'}$.

Dass während des Algorithmus kein Angebot von i' an j' erfolgte, kann somit nur daran liegen, dass $a_{i'j'} = c_{i'j'} = 0$ gilt. Denn nur so können die bisherigen Schritte des Algorithmus abgelaufen sein, ohne dass ein Angebot von i' an j' unterbreitet wurde.

Aus dem Grund gehen wir beim Matching der freien Kapazitäten an dieser Stelle wie folgt vor: Wir matchen die freien Kapazitäten der Agenten aus P und Q willkürlich. Dabei werden rigide vor flexiblen Kanten bevorzugt. Wie beim Beweis des Algorithmus gezeigt wird, erhalten wir dadurch ein paarweise stabiles Outcome.

Wir setzen noch zum Abschluss für alle $1 \leq i \leq n$, $1 \leq j \leq m$

$$u^R_{ij} = u^R_{\sigma,v}(i,j)$$

und

$$u_{ij}^F = u_{\sigma,v}^F(i,j)$$

und sind damit fertig.

5.4 Pseudocodes

Im folgenden werden die Pseudocodes des b-HNS-Algorithmus und seiner Unteralgorithmen dargestellt. Einzelne kleinere Prozeduren werden dabei nicht im Detail aufgeführt. Man beachte hierzu auch die Erläuterungen zu einzelnen Prozeduren in Kapitel 5.3.

5.4.1 Der b-HNS-Algorithmus

$V^R \leftarrow 0$
$V^F \leftarrow 0$
$N \leftarrow 0$
$P_{cap}^{Anf} \leftarrow P_{cap}$
PlaceProposals
Konstruiere den Digraphen $G^{\sigma,v}$
while es gibt ein $j_0 \in Q$, das überschüssige Kapazitäten hat **do**
  while es gibt ein Netzwerk $\mathcal{N}$, der die Quelle j_0 besitzt und als Senke eine insolvente Firma besitzt oder ein $i_0 \in P$ und i_0 ist in $G^{\sigma,v}$ Anfangsknoten eines rigiden Pfeils oder einen Knoten j_1 aus Q, der noch offene Kapazitäten besitzt oder bereits ein rigides Angebot **do**
    ScalingMaxFlow($\mathcal{N}$)
    CheckProposals
    DisposeRigid (j_1)
    Alternate ($\mathcal{N}$)
    PlaceProposals
    Aktualisiere $G^{\sigma,v}$, K^R und K^F
  end while
  HungarianUpdate
  Aktualisiere $G^{\sigma,v}$, K^R und K^F
end while
while es gibt eine Firma $i \in P$, für die $\sum_{j\in Q} \sigma_{ij}^R + \sum_{j\in Q} \sigma_{ij}^F < p_i^{Anf}$ gilt, und es gibt ein $j \in Q$ mit $\sigma_{ij}^R < k_{ij}^R$ oder $\sigma_{ij}^F < k_{ij}^F$ und j besitzt noch freie Kapazitäten **do**
  if es gibt ein $j \in Q$ mit dem eine rigide Kante gebildet werden kann **then**
    $\sigma_{ij}^R \leftarrow \min\left\{p_i^{Anf} - \left(\sum_{t\in Q}\sigma_{it}^F + \sum_{t\in Q}\sigma_{it}^R\right), q_j^* - \left(\sum_{s\in P}\sigma_{sj}^F + \sum_{s\in P}\sigma_{sj}^R\right), k_{ij}^R\right\}$
    $v_{ij}^R \leftarrow b_{ij}$
  else wähle ein $j \in Q$ mit dem eine flexible Kante gebildet werden kann
    $\sigma_{ij}^F \leftarrow \min\left\{p_i^{Anf} - \left(\sum_{t\in Q}\sigma_{it}^F + \sum_{t\in Q}\sigma_{it}^R\right), q_j^* - \left(\sum_{s\in P}\sigma_{sj}^F + \sum_{s\in P}\sigma_{sj}^R\right), k_{ij}^F\right\}$
    $v_{ij}^F \leftarrow c_{ij}$

26: **end if**
27: **end while**
28: **for all** $i \in P$ **do**
29: **for all** $j \in Q$ **do**
30: $u_{ij}^R \leftarrow u_{\sigma,v}^R(i,j)$
31: $u_{ij}^F \leftarrow u_{\sigma,v}^F(i,j)$
32: **end for**
33: **end for**

5.4.2 PlaceProposals

1: **procedure** PlaceProposals
2: **while** es gibt eine Firma $i \in P$, die noch solvente, freie Kapazitäten besitzt **do**
3: **while** es gibt eine Firma $i \in P$, die noch solvente, freie Kapazitäten besitzt **do**
4: Propose (i)
5: **end while**
6: **for all** $j \in Q$ **do**
8: **for all** $j \in P$ **do**
9: **if** $X = R$ in $(i,j)^X$ **then**
10: $v_{ij}^R \leftarrow b_{ij}$
11: **end if**
12: **end for**
13: Konstruiere Z_j, Z_j^*, Z_j'
14: **if** $\sum_{i=1}^{n} \sigma_{ij}^R + \sum_{i=1}^{n} \sigma_{ij}^F \geq q_j^*$ **then**
15: Definiere $\left(\widetilde{i}, \sigma_{\widetilde{i},j}^X\right)$
16: **if** es gilt $X = R$ in $\left(\widetilde{i}, \sigma_{\widetilde{i},j}^X\right)$ **then**
17: $\left(\widetilde{i}, \sigma_{\widetilde{i},j}^R\right) \leftarrow \left(\widetilde{i}, \tau_{\widetilde{i},j}^R\right)$
18: **end if**
19: **end if**
20: **if** $\sum_{i=1}^{n} \sigma_{ij}^R + \sum_{i=1}^{n} \sigma_{ij}^F > q_j^*$ **then**
21: CheckProposals
22: **end if**
23: **end for**
24: **end while**
25: **end procedure**

5.4.3 Propose(x)

1: **procedure** Propose
2: Sortiere die Kanten $(i,j)^X$, $\forall j \in Q, X \in \{R,F\}$ nach dem Profit für x. Bei gleichem Profit stehen rigide vor flexiblen Kanten.
3: Ermittle D_i

4: Ermittle d_i
5: Vergib Kapazitäten auf Kanten aus D_i, so dass der Profit von i maximiert wird. Vergib die Kapazitäten in der ermittelten Reihenfolge.
6: **end procedure**

5.4.4 CheckProposals

1: **procedure** CheckProposals
2: **for all** $\left(i', \sigma^R_{i'j}\right) \in Z'_j$ mit $\left(i', \sigma^R_{i'j}\right) \neq \left(\widetilde{i}, \sigma^X_{\widetilde{i}j}\right)$ **do**
3: $p^*_{i'} \leftarrow p^*_{i'} + \sigma^R_{i'j}$
4: $\sigma^R_{i'j} \leftarrow 0$
5: $Z_j \leftarrow Z_j \setminus \left(i', \sigma^R_{i'j}\right)$
6: $Z'_j \leftarrow Z'_j \setminus \left(i', \sigma^R_{i'j}\right)$
7: **end for**
8: **if** $X = R$ in $\left(\widetilde{i}, \sigma^X_{\widetilde{i}j}\right)$ und $Z'_j \geq 2$ **then**
9: **for all** $i \in P$ **do**
10: **if** $\left(i, \tau^F_{ij}\right) \notin Z^*_j$ **then**
11: $v^F_{ij} \leftarrow \min\left\{v^R_{\widetilde{i}j}, c_{ij}\right\}$
12: Aktualisiere u^F_{ij}
13: **if** $u^F_{ij} \leq d_i$ **then**
14: $p^*_{i'} \leftarrow p^*_{i'} + \sigma^F_{i'j}$
15: $\sigma^F_{i'j} \leftarrow 0$
16: **end if**
17: **end if**
18: **if** $\left(i, \tau^R_{ij}\right) \notin Z^*_j$ **then**
19: $v^R_{ij} \leftarrow v^R_{\widetilde{i}j}$
20: **end if**
21: **end for**
22: Konstruiere Z_j, Z^*_j, Z'_j, $\left(\widetilde{i}, \sigma^X_{\widetilde{i}j}\right)$ neu
23: **for all** $\left(i', \sigma^R_{i'j}\right) \in Z'_j$ **do**
24: **if** $\left(i', \sigma^R_{i'j}\right) \neq \left(\widetilde{i}, \sigma^X_{\widetilde{i}j}\right)$ **then**
25: $p^*_{i'} \leftarrow p^*_{i'} + \sigma^R_{i'j}$
26: $\sigma^R_{i'j} \leftarrow 0$
27: $Z_j \leftarrow Z_j \setminus \left(i', \sigma^R_{i'j}\right)$
28: $Z'_j \leftarrow Z'_j \setminus \left(i', \sigma^R_{i'j}\right)$
29: **end if**
30: **end for**
31: **end if**
32: **end procedure**

5.4.5 HungarianUpdate

1: **procedure** HungarianUpdate
2: Wähle $j \in Q$, das überschüssige Kapazitäten besitzt

3: Bestimme die Knoten $\overline{P} \subseteq P$ und $\overline{Q} \subseteq Q$, die in $G^{\sigma,v}$ in der Komponente von j liegen
4: **for all** $i \in \overline{P}$ **do**
5: $u_i \leftarrow \max_{k=1}^{m} \max_{X \in \{R,F\}} u^X_{\sigma,v}(i,k)$
6: **end for**
7: $\Delta_1 \leftarrow \min \left\{ u_i - \max_{X \in \{R,F\}} u^X_{\sigma,v}(i,j) \,|\, i \in \overline{P}, j \notin \overline{Q} \right\}$
8: $\Delta_2 \leftarrow \min \left\{ u_i - u^R_{\sigma,v}(i,j) \,|\, i \in \overline{P}, j \in \overline{Q} \right\}$
9: $\Delta_3 \leftarrow \min \{u_i\}$
10: $\Delta \leftarrow \min \{\Delta_1, \Delta_2, \Delta_3\}$
11: **for all** $j' \in \overline{Q}$ **do**
12: **for all** $i' \in \overline{P}$ **do**
13: $v^F_{i'j'} \leftarrow v^F_{i'j'} + \Delta$
14: **end for**
15: **end for**
16: **end procedure**

5.5 Beweis der Korrektheit

Bei dem folgenden Beweis orientiere ich mich in den grundlegenden Zügen am Beweis aus [13]. Die Zeilenangaben in diesem Kapitel beziehen sich jeweils auf die Pseudocodes der Algorithmen.

Zuerst ist festzustellen, dass alle Anweisungen durchgeführt werden können. Insbesondere kann jede Firma i, für die PROPOSE(i) aufgerufen wird, auch tatsächlich ausreichend Angebote unterbreiten. Dies wird bereits dadurch sichergestellt, dass nur solvente Kapazitätsanteile PROPOSE(i) durchlaufen.

Proposition 5.5.1

Sei $j \in Q$. Bei Durchführung von CHECKPROPOSALS *ändert sich die Anzahl der j angebotenen Kapazitäten nicht, falls* $\sum_{i=1}^{n} \sigma^R_{ij} + \sum_{i=1}^{n} \sigma^F_{ij} \leq q^*_j$.

Beweis
Sei $j \in Q$ ein Knoten mit $\sum_{i=1}^{n} \sigma^R_{ij} + \sum_{i=1}^{n} \sigma^F_{ij} \leq q^*_j$. Dann ist Z'_j entweder leer oder besteht aus genau einem Element. Falls Z'_j leer ist, löscht CHECKPROPOSALS keine Kapazitäten von j. Andernfalls löscht CHECKPROPOSALS auch keine Kapazitäten von j, weil weder die for-Schleife ab Zeile 2 durchlaufen wird, noch die if-Bedingung in Zeile 8 erfüllt ist. □

Proposition 5.5.2

Werden von DISPOSERIGID *einem Knoten $j \in Q$ nicht überflüssige Kapazitäten entzogen, so werden diese Kapazitäten sofort wieder durch* ALTERNATE *aufgefüllt.*

Beweis
Mittels DISPOSERIGID werden von einem Knoten $j \in Q$ nicht überflüssige Kapazitäten nur dann gestrichen, wenn in $G^{\sigma,v}$ ein Fluss von einem Knoten aus Q mit überschüssigen Kapazitäten zu j existiert. Die Anzahl der gestrichenen Kapazitäten ist nicht größer als die Kapazität des Flusses in $G^{\sigma,v}$ zu j. Nachdem die Kapazität des Flusses aber auch durch die Kapazität der Senke j begrenzt wird und diese eben über die Menge der vorliegenden rigiden Kapazitäten bestimmt wird, werden höchstens so viele rigide Kanten gelöscht, wie im Anschluss durch ALTERNATE wieder aufgefüllt werden. □

Proposition 5.5.3

Sei für alle $j \in Q$ cap_j definiert als $cap_j := q_j^*$, falls $\sum_{i=1}^{n} \sigma_{ij}^R + \sum_{i=1}^{n} \sigma_{ij}^F \geq q_j^*$ und $cap_j := \sum_{i=1}^{n} \sigma_{ij}^R + \sum_{i=1}^{n} \sigma_{ij}^F$ sonst. Dann wird cap_j und somit auch $\sum_{j \in Q} cap_j$ niemals durch PLACEPROPOSALS *reduziert.*

Beweis
Für ein beliebiges $j \in Q$ wird cap_j höchstens durch PROPOSE oder durch CHECKPROPOSALS geändert. Für PROPOSE gilt die Behauptung offensichtlich, für CHECKPROPOSALS folgt die Behauptung aus Proposition 5.5.1. □

Proposition 5.5.4

Für alle $i \in P$, $j \in Q$ gilt, dass v_{ij}^R und v_{ij}^F während des gesamten Algorithmus nicht kleiner werden.

Beweis
Wir untersuchen die Stellen in den Algorithmen, wo sich Änderungen an den Vektoren V^R und V^F ergeben können. In der Prozedur PROPOSE werden die Elemente von $V^{R/F}$ nie reduziert. Denn ein Angebot wird von einem Agenten $i \in P$ an ein $j \in Q$ höchstens dann unterbreitet, wenn nicht nur der Profit von i dadurch steigt, sondern wenn dadurch auch v_{ij}^R und v_{ij}^R nicht sinken. Beim ersten Aufruf von PROPOSE ist dies dadurch sichergestellt, dass V^R und V^F Nullmatrizen sind. Bei einem späteren Aufruf von PROPOSE definieren $u_{\sigma,v}^{R/F}$ den zu erwartenden Profit von i bei einem Angebot an j. Dieser hängt von $v_{ij}^{R/F}$ ab und ist 0, falls ein Angebot von i an j zu einer Verschlechterung des Profits von j führen würde. Solche Angebote werden dann nicht unterbreitet.

Wird in Zeile 22 oder Zeile 25 des b-HNS-Algorithmus einem v_{ij}^R beziehungsweise v_{ij}^F ein $b_{ij} \geq 0$ beziehungsweise $c_{ij} \geq 0$ zugeordnet, dann geschieht das nur, wenn j noch freie Kapazitäten besitzt. Wenn dies der Fall ist, dann folgt aus Proposition 5.5.2, dass j während des gesamten Algorithmus noch nie ausreichend viele Angebote vorliegen hatte, dass also noch nie $cap_j = q_j^*$ galt. Das wiederum ist jedoch Voraussetzung dafür, dass für eine bislang nicht bestehende Kante $(i,j)^{R/F}$ v_{ij}^R oder v_{ij}^F größer als 0 ist. Denn nur, wenn für j $cap_j = q_j^*$

gilt, werden auch Profite $v_{ij}^{R/F}$ geändert, obwohl die Kante $(i,j)^{R/F}$ gar nicht besteht. Somit gilt $v_{ij}^R = 0$ bzw. $v_{ij}^R = 0$ und die Behauptung stimmt für diese Stelle des Algorithmus.

Nachdem, wie beschrieben, Angebote nur unterbreitet werden, wenn der Profit von $j \in Q$ dadurch nicht sinkt, stimmt die Behauptung auch für die Anweisung in Zeile 10 von Algorithmus PlaceProposals.

Da $\left(i, \tau_{ij}^F\right) \notin Z_j^*$, gilt in Zeile 11 von Algorithmus CheckProposals, dass $v_{ij}^F \leq v_{\widetilde{ij}}^R$. Da weiterhin immer gilt, dass $v_{ij}^F \leq c_{ij}$, folgt die Behauptung für diese Stelle des Algorithmus.

Analog folgt aus $\left(i, \tau_{ij}^R\right) \notin Z_j^*$ für die Zuweisung in Zeile 19 von Algorithmus CheckProposals, dass $v_{ij}^R \leq v_{\widetilde{ij}}^R$. Deswegen reduziert die Zuweisung $v_{ij}^R \leftarrow v_{\widetilde{ij}}^R$ den Wert von v_{ij}^R nicht.

In Zeile 13 des HungarianUpdate ist $\Delta > 0$, was in 5.5.7 gezeigt wird, und die Behauptung stimmt auch hier. □

Proposition 5.5.5

Beim Aufruf von HungarianUpdate *sei $j \in Q$ ein Knoten, für den $\sum_{i=1}^{n} \sigma_{ij}^R + \sum_{i=1}^{n} \sigma_{ij}^F > q_j^*$ gilt. Sei $(i,j)^X, X \in \{R,F\}$ die Kante, die j zumindest teilweise zur Abdeckung seiner Kapazitäten bevorzugt, die aber von j von allen solchen Kanten am wenigsten bevorzugt wird. Dann ist die Kante $(i,j)^X$ flexibel, das heißt es gilt $X = F$. Außerdem gehören alle überschüssigen Kapazitäten von j zu flexiblen Kanten.*

Beweis
Bei der Kante $(i,j)^X$ handelt es sich um die Kante, die zum oben definierten Tupel $\left(\widetilde{i}, \sigma_{\widetilde{ij}}^X\right) \in Z_j^*$ gehört, das heißt es gilt $\widetilde{i} = i$. Vor Aufruf des HungarianUpdate wird PlaceProposals und somit insbesondere auch wegen $\sum_{i=1}^{n} \sigma_{ij}^R + \sum_{i=1}^{n} \sigma_{ij}^F > q_j^*$ CheckProposals ausgeführt. Dort werden aber alle rigiden Kanten, die zu einem Tupel aus $Z_j' \setminus \left\{\left(\widetilde{i}, \sigma_{\widetilde{ij}}^X\right)\right\}$ gehören, gelöscht. Vor Aufruf des HungarianUpdate ist daher sichergestellt, dass alle überschüssigen Kapazitäten zu flexiblen Kanten gehören.

Die Kante $(i,j)^X$ selbst muss flexibel sein. Denn andernfalls greift die if-Bedingung in Zeile 8 von CheckProposals. Nach deren Ausführung ist entweder $\sum_{i=1}^{n} \sigma_{ij}^R + \sum_{i=1}^{n} \sigma_{ij}^F = q_j^*$ oder es wird eine neue Kante relevant und diese ist notwendig flexibel. □

Proposition 5.5.6

Sei $\overline{P}$ wie im HungarianUpdate *für ein passendes $j \in Q$ definiert. Alle Kanten in $G^{\sigma,v}$, die in einem $i \in \overline{P}$ enden, sind flexibel.*

Beweis
Sei $(i, j')^{R/F}$ eine Kante mit $i \in \overline{P}$. Angenommen, es ist $j' = j$. Dann folgt aus Proposition 5.5.5, dass j' nur flexible Kanten besitzt und somit insbesondere die Kante zu i flexibel ist.

Sei nun $j' \in Q$ mit $j' \neq j$. Aus der Konstruktion von $\overline{P}$ folgt, dass es ein Netzwerk geben muss mit Quelle j, das auch die Kante $(i, j')^R$ beinhaltet. Dann hätte aber die while-Schleife ab Zeile 7 vom b-HNS-Algorithmus nicht verlassen werden dürfen.

Daraus folgt die Behauptung. □

Satz 5.5.7

Der b-HNS-Algorithmus terminiert nach endlich vielen Schritten.

Beweis
Wir müssen zeigen, dass die einzelnen while-Schleifen jeweils nur endlich oft durchlaufen werden. Bei allen anderen Anweisungen ist dies offensichtlich.

Die Anzahl der Durchläufe der inneren while-Schleife von PLACEPROPOSALS ab Zeile 3 wird für jede gegebene Ausgangssituation durch die Anzahl der Elemente in P begrenzt. Auch die äußere while-Schleife ab Zeile 2 wird nur endlich oft durchlaufen. Zwar kann es sein, dass gemachte Angebote gestrichen oder reduziert werden, aber solche Angebote werden nicht wieder unterbreitet bzw. bei solchen Angeboten werden bestehende Kapazitäten nicht mehr erweitert. Dafür sorgen zum einen die Definitionen der Funktionen $u^R_{\sigma,v}$ und $u^F_{\sigma,v}$, zum anderen auch die Matrix N. Da die Anzahl möglicher Angebote endlich ist, wird diese while-Schleife auch nur endlich oft durchlaufen.

In der inneren while-Schleife ab Zeile 8 des b-HNS-Algorithmus werden entweder neue rigide Angebote unterbreitet, bestehende rigide Angebote gelöscht, Kapazitäten einer Firma als insolvent markiert oder es werden Kapazitäten von einem Knoten mit überflüssigen Kapazitäten zu einem Knoten mit Kapazitätsbedarf alterniert. Dies ist nur endlich oft möglich, da die Anzahl der Agenten und die Summe aller Kapazitäten endlich sind.

Um zu zeigen, dass auch die äußere while-Schleife ab Zeile 7 des b-HNS-Algorithmus while-Schleife nur endlich oft durchlaufen wird, überlegen wir Folgendes:

Falls keine Netzwerke $\mathcal{N}$ mehr vorliegen, die eine dieser vier Möglichkeiten bietet, wird HUNGARIANUPDATE aufgerufen. Mit Proposition 5.5.6 sind alle Knoten von $\overline{P} \cup \overline{Q}$ in $G^{\sigma,v}$ ausschließlich inzident zu flexiblen Pfeilen. Daraus folgt, dass $\Delta_2 > 0$ ist. Gemäß Konstruktion sind bei Aufruf von HUNGARIANUPDATE auch Δ_1 und Δ_3 größer als 0, woraus $\Delta > 0$ folgt.

Angenommen, es gibt ein Tripel (X, i_0, z_0), so dass $\Delta = \Delta_1 = u_{i_0} - u^X_{\sigma,v}(i_0, z_0)$ und $(i_0, z_0) \notin D^{i_0}_{\sigma,v}$. Wir zeigen, dass durch das Update in den Zeilen 11 bis 15 des Algorithmus HUNGARIANUPDATE die Kante (i_0, z_0) in $G^{\sigma,v}$ aufgenommen wird, gleichzeitig aber keine bestehende Kante aus $G^{\sigma,v}$ gestrichen wird.

Δ ist so konstruiert, dass keine neue Kante von einem Knoten aus P einer bislang bestehenden Kante vorgezogen wird. Allenfalls werden nun weitere Kanten als ebenso profitabel für eine Firma angesehen. Somit werden in $G^{\sigma,v}$ keine rückwärts gerichteten Pfeile gelöscht.

Sei nun $(i,j)^X, X \in \{R,F\}$ ein vorwärts gerichteter Pfeil in $G^{\sigma,v}$. Dann erfüllt er die Bedingungen

$$u^X_{\sigma,v}(i,j) = \max_{k=1}^{m} \max_{X \in \{R,F\}} u^X_{\sigma,v}(i,k) > 0 \tag{5.25}$$

und

$$\sigma^X_{ij} < k^X_{ij}. \tag{5.26}$$

Die Gleichung (5.25) wird durch Δ beeinflusst. Denn nachdem v^F_{ij} (es handelt sich um eine flexible Kante zwischen i und j) um Δ erhöht wird, sinkt wegen $u^F_{\sigma,v}$ der Profit von i an dieser Kante. Nachdem der gleiche Betrag zu allen Knoten in $\overline{Q}$ hinzugefügt wird und i in $G^{\sigma,v}$ nur zu Knoten aus $\overline{Q}$ adjazent ist, bleibt (5.25) weiterhin gültig. Damit ist $(i,j)^F \in D^i_{\sigma,v}$ und wird somit nicht gestrichen.

Nachdem wie bereits oben gezeigt eine einmal reduzierte oder gelöschte rigide Kante nicht wieder weitere Kapazitäten erhält, ist die Anzahl der Schritte, bei denen rigide Kanten aufgenommen werden, begrenzt. Ebenso bleiben einmal als insolvent gekennzeichnete Kapazitäten einer Firma immer insolvent, so dass auch hier nur endlich viele solcher Schritte möglich sind. Nachdem im Fall $\Delta = \Delta_2$ eine neue Kante in den Digraphen aufgenommen wird und bei $\Delta = \Delta_3$ Kapazitäten einer Firma als insolvent gekennzeichnet werden, wird in diesen Fällen die while-Schleife nur endlich oft durchlaufen.

Die while-Schleife ab Zeile 19 des b-HNS-Algorithmus terminiert offensichtlich auch nach endlich vielen Schritten. Denn solange die Voraussetzungen der while-Schleife erfüllt sind, wird bei jedem Durchlauf mindestens eine Kapazität neu gematcht. Außerdem ist die Anzahl der Durchläufe durch das Produkt der Anzahl der Knoten von P und derer von Q begrenzt. Nachdem sowohl die Anzahl der Kapazitäten als auch die Anzahl der Agenten endlich ist, wird die while-Schleife nur endlich oft ausgeführt.

Somit terminiert der Algorithmus nach endlich vielen Schritten. □

Satz 5.5.8

Der erweiterte HNS-Algorithmus erzeugt ein zulässiges, insbesondere aber auch paarweise stabiles Outcome.

Beweis
Wir zeigen der Reihe nach, dass das Outcome nach Ende des Algorithmus die bei der Modellierung des Problems geforderten Eigenschaften besitzt.

Eine Firma gibt über PlaceProposals immer nur so viele Kapazitäten in seine Angebote, wie sie solvente Kapazitäten besitzt. An anderen Stellen des Algorithmus werden höchstens Kapazitäten der Firmen wieder freigegeben. Eine Alternierung ändert nichts an der Zahl der vergebenen Kapazitäten einer Firma. Somit ist (5.1) für alle $i \in P$ erfüllt.

Die Ungleichungen in (5.2) folgen aus Satz 5.5.7. In den while-Schleifen des b-HNS-Algorithmus werden überschüssige Kapazitäten eines Arbeiters sukzessive

reduziert. Nachdem der Algorithmus terminiert, muss also dann eine Situation vorliegen, in der kein Arbeiter mehr überschüssige Kapazitäten besitzt.

Eine Änderung des Wertes v_{ij}^R erfolgt an allen Stellen des Algorithmus stets unter Beachtung der Gleichheit gemäß (5.4). Bei Erhöhung des Profits v_{ij}^F aus einer flexiblen Kante wird während des Algorithmus stets berücksichtigt, dass $v_{ij}^F \leq c_{ij}$ ist. Damit ist sichergestellt, dass die Gleichung in (5.4) eingehalten werden kann, da U^F nur nichtnegative Einträge besitzt. $u_{\sigma,v}^F$ und $u_{\sigma,v}^R$ sind gerade so konstruiert, dass die Aussagen (5.3) und (5.4) zutreffen, wenn die Zuweisung der Profite an $u_{ij}^{R/F}$ am Ende des Algorithmus erfolgt.

Zu Beginn des Algorithmus besitzt jede Kante die Kapazität 0. Hier treffen die Aussagen (5.5) und (5.6) ganz natürlich zu. In PlaceProposals werden Angebote nur unter Berücksichtigung der Kapazitätsrestriktionen der einzelnen Kanten unterbreitet. Vereinzelt werden im Unteralgorithmus CheckProposals noch Kapazitäten gestrichen. Nach Ausführung von PlaceProposals sind daher in einem Graphen, in dem die Ungleichungen aus (5.5) und (5.6) vor der Ausführung von PlaceProposals erfüllt waren, auch weiterhin (5.5) und (5.6) erfüllt. Auch das HungarianUpdate führt nicht zu einer Überschreitung der Kapazitätsrestriktionen bei den Kanten. In der inneren while-Schleife des b-HNS-Algorithmus können neue Kanten entstehen und zu bestehenden Kanten Kapazitäten hinzugefügt werden. Die geschieht durch Alternierungen. Die Kapazität eines vorwärts gerichteten Pfeils wird dabei zu den Kapazitäten der zugehörigen Kante hinzugefügt. Die Definition der Pfeilkapazitäten in (5.18) und (5.19) führt nun dazu, dass durch diese Alternierungen die Kapazitätsrestriktionen der Kanten nicht überschritten werden. Auch bei der abschließenden Verteilung der als insolvent markierten Kapazitäten von Firmen werden die Kapazitätsrestriktionen der Kanten berücksichtigt.

Somit gelten auch die Ungleichungen in (5.5) und (5.6). Und damit ist das Outcome, das durch den b-HNS-Algorithmus erzeugt wird, zulässig.

Es bleibt zu zeigen, dass auch die Bedingungen (5.7), (5.8) und (5.9) erfüllt werden und somit vom b-HNS-Algorithmus ein paarweise stabiles Outcome erzeugt wird.

Wir beweisen dazu zuerst, dass die Aussage (5.7) zutrifft. Seien dazu $i \in P$ und $j \in Q$ zwei Agenten, die jeweils noch nach Abschluss des Algorithmus noch freie Kapazitäten besitzen. Damit sind die Voraussetzungen von (5.7) erfüllt.

Unabhängig vom Verlauf des Algorithmus werden am Ende in der while-Schleife ab Zeile 19 alle Agenten überprüft und freie Kapazitäten nach Möglichkeit gematcht. An dieser Stelle spielen dann Profite aus diesen Matchings keine Rolle mehr. Wenn der Algorithmus also beendet ist, dann muss notwendig für i und j $\sigma_{ij}^R = k_{ij}^R$ und $\sigma_{ij}^F = k_{ij}^F$ gelten. Andernfalls hätte die while-Schleife nicht verlassen werden dürfen.

Für den Beweis der Aussage (5.8) seien $i' \in P$ und $j' \in Q$ zwei Agenten, für die $\sigma_{i'j'}^R < k_{i'j'}^R$ nach Abschluss des Algorithmus gilt. Falls $p_{i'}^* = \sigma_{i'j'}^R$ gilt, dann ist $u_{\min,j'}^R(i') = u_{i'j'}^R = a_{i'j'}$ und die Aussage ist wahr. Analog folgt aus $q_{j'}^* = \sigma_{i'j'}^R$, dass $v_{\min,i'}^R(j') = v_{i'j'}^R = b_{i'j'}$. Wir können daher für den weiteren Beweis dieser Aussage annehmen, dass i' und j' jeweils noch zu weiteren Kanten neben $(i',j')^R$ inzident sind.

Die Aussage (5.8) ist bewiesen, wenn wir zeigen können, dass aus

$$a_{i'j'} > u_{\min,j'}^R(i') \tag{5.27}$$

folgt, dass

$$b_{i'j'} \leq v^R_{\min,i'}(j') \tag{5.28}$$

gilt. Angenommen i' erfüllt (5.27). Sei $(i', j'')^X, X \in \{R, F\}$ mit $\sigma^X_{i',j''} > 0$ eine Kante aus dem Outcome des Algorithmus für die $u^X_{i',j''} = u^R_{\min,j'}(i')$ gilt. In PLACEPROPOSALS würde i' erst an j' ein Angebot unterbreiten, ehe ein Angebot an j'' erfolgt. Dass dennoch ein Angebot an j'' gemacht wurde, kann nur daran liegen, dass $n_{i'j'} = 1$ ist, da die andere Möglichkeit, dass $\sigma^R_{i'j'} < k^R_{i'j'}$ gilt, in den Voraussetzungen ausgeschlossen wurde. $n_{i'j'} = 1$ gilt wiederum nur dann, wenn j' ausreichend Angebote vorliegen und $v^R_{i'j'}$ so niedrig ist, dass j' keine oder nicht alle Kapazitäten aus der Kante $(i', j')^R$ zur profitmaximierenden Deckung seines Kapazitätsbedarfs heranzieht. Das heißt aber wiederum, da die Profite von j' gemäß Proposition 5.5.4 während des Algorithmus nicht kleiner werden, dass $b_{i'j'} \leq v^R_{\min,i'}(j')$ gelten muss.

Zum Beweis der Aussage (5.9) betrachten wir nun beliebige $i' \in P$ und $j' \in Q$, für die $\sigma^F_{i'j'} < k^F_{i'j'}$ gilt. Wie beim Beweis von (5.8) können wir annehmen, dass beide Agenten nicht bereits ihre jeweils zur Verfügung stehenden Kapazitäten voll in die Kante $(i', j')^F$ gegeben haben. Denn dann folgt die Aussage sofort.

Der Algorithmus CHECKPROPOSALS sorgt dafür, dass der mögliche Profit aus einer erweiterbaren flexiblen Kante (i', j') für j' so hoch ist wie der minimale Profit der anderen bestehenden Kanten. Es gilt also die erste wichtige Feststellung:

$$v^F_{i'j'} = v^F_{\min,i'}(j'). \tag{5.29}$$

Angenommen, es existiert im Outcome des Algorithmus eine Kante $(i', j'')^X, X \in \{R, F\}$ mit $\sigma^X_{i'j''} > 0$, für die $u^X_{i'j''} < u^F_{i'j'}$ gilt.

Nehmen wir zuerst an, es ist $X = R$. Nachdem mit Proposition 5.5.4 der Profit einer flexiblen Kante während des Algorithmus für den Partner aus P nicht steigt, war der Profit für i' aus der Kante $(i', j')^F$ während des gesamten Algorithmus größer als der Profit aus der Kante$(i', j'')^R$. Während des Algorithmus werden nur rigide Kanten von Arbeitern abgelehnt und für i' gibt es mit dem eben Gesagten keinen Anlass, $(i', j')^F$ zu reduzieren, aber die Kapazitäten in $(i', j'')^R$ bestehen zu lassen. Daher kann $\sigma^F_{i'j'} < k^F_{i'j'}$ nur durch Alternierungen entstanden sein. Die durch die Alternierung freigewordenen Kapazitäten werden in eine neue Kante gegeben. Der Profit für i' aus der neuen Kante ist gleich dem Profit aus $(i', j')^F$. Das wiederum heißt aber, dass eine Kante existiert, die einen höheren Profit für i' ergibt als die Kante $(i', j'')^R$. Dann hätte aber i' kein Angebot an j'' unterbreitet. Wir erhalten einen Widerspruch und erkennen, dass $X = R$ nicht gelten kann.

Nehmen wir daher nun an, dass $X = F$ gilt. Falls während des gesamten Algorithmus gilt, dass der Profit für i' aus der flexiblen Kante zu j'' kleiner als der Profit der flexiblen Kante zu j' ist, dann können wir wie im Fall $X = R$ argumentieren und erkennen, dass dieser Fall nicht auftreten kann. Nehmen wir nun an, dass zu einem Zeitpunkt im Algorithmus der Profit der flexiblen Kante zu j'' größer als der Profit der flexiblen Kante zu j' war. In diesem Fall kann durchaus ein Angebot von i' an j'' unterbreitet worden sein. Wir betrachten hier den Fall, dass es sich um ein flexibles Angebot handelte. Die Tatsache, dass nach Abschluss $u^F_{i'j''} < u^F_{i'j'}$, kann nur daraus entstanden sein, dass der Profit

aus dieser Kante für j'' während des Algorithmus entsprechend mehr stieg als der Profit für j' aus der Kante $(i', j')^F$. Hier sorgt aber die if-Bedingung in Zeile 13 von CheckProposals dafür, dass eine maximale Zahl von Kapazitäten von der Kante $(i', j'')^F$ gestrichen wird, die in Kanten mit höherem Profit für i' untergebracht werden können. Das bedeutet, dass die Kante $(i', j'')^F$ entweder leer ist, oder es existieren keine Möglichkeiten für i' mehr, weitere Kapazitäten in Kanten mit höherem Profit zu geben. Beide Aussagen stehen aber im Widerspruch zu unserer Annahme über die Kante $(i', j'')^F$. Somit ist gilt für X auch $X \neq F$. Daraus folgt aber wiederum, dass es im Outcome des Algorithmus keine Kante $(i', j'')^X, X \in \{R, F\}$ mit $\sigma^X_{i'j''} > 0$, für die $u^X_{i'j''} < u^F_{i'j'}$ gilt, gibt.

Und somit gilt die zweite wichtige Feststellung:

$$u^F_{i'j'} \leq u^F_{\min,j'}(i'). \tag{5.30}$$

Aus (5.29) und (5.30) folgt damit aber für beliebige $i' \in P$, $j' \in Q$

$$c_{i'j'} = u^F_{i'j'} + v^F_{i'j'} \leq u^F_{\min,j'}(i') + v^F_{\min,i'}(j') \tag{5.31}$$

und somit die Aussage (5.9). Und damit haben wir gezeigt, dass der b-HNS-Algorithmus ein paarweise stabiles Outcome erzeugt.

5.6 Komplexitätsanalyse

Der b-HNS-Algorithmus ist an den HNS-Algorithmus angelehnt, der in $\mathcal{O}(n^4)$ abläuft, wobei dort $|P| = |Q| = n$ gilt und jeder Agent genau eine Arbeitszeiteinheit zu vergeben hat. Die folgende Analyse lehnt sich in weiten Teilen an die Komplexitätsanalyse dort an. Bei unserem neuen Algorithmus müssen wir berücksichtigen, dass jeder Knoten eine individuelle Kapazität besitzen kann. Wir definieren

$$k := \max\left\{k^X_{ij} | i \in P, j \in Q, X \in \{R, F\}\right\}$$

als die maximale Kapazität einer Kante, die im b-Matching vorkommen kann. Beim HNS-Algorithmus kann man die Aussage treffen, dass nach Abschluss des Algorithmus n Kanten existieren. Diese Aussage lässt sich nicht so einfach auf den b-HNS-Algorithmus übertragen. Die Anzahl der möglichen Kanten hängt von n, m und k an. Man kann die Aussage treffen, dass in jeder zulässigen Lösung des b-HNS-Algorithmus höchstens $\min\{n \cdot m, k \cdot n, k \cdot m\}$ Kanten vorliegen.

Wir untersuchen zuerst den Algorithmus Propose. Eine Sortierung der $2m$ möglichen zu i inzidenten Kanten kann mit einem geeigneten Sortieralgorithmus in $\mathcal{O}(\log m)$ durchgeführt werden (vergleiche [11]). Die Ermittlung von D_i und d_i kann wie die Verteilung der Kapazitäten auf die Kanten in Linearzeit $\mathcal{O}(m)$ ablaufen. Somit hat Propose eine Laufzeit von $\mathcal{O}(m)$.

Betrachten wir weiter CheckProposals. Die for-Schleife in den Zeilen 2 bis 7 kann in Linearzeit $\mathcal{O}(n)$ ausgeführt werden. Die Laufzeit der if-Bedingung ab Zeile 8 hängt von den beiden for-Schleifen ab Zeile 9 beziehungsweise ab Zeile 23 ab. Die Konstruktionen in Zeile 22 laufen in $\mathcal{O}(\log n)$ ab (analog zu oben, siehe [11]), wenn man einen entsprechenden Sortieralgorithmus verwendet. Die for-Schleife ab Zeile 23 benötigt $\mathcal{O}(n)$. Die for-Schleife ab Zeile 9 kann höchstens n-mal aufgerufen werden. Alle Zuordnungen und Entscheidungen hierin benötigen $\mathcal{O}(1)$ Laufzeit. Somit besitzt der gesamte Algorithmus CheckProposals eine Laufzeit von $\mathcal{O}(n)$.

Mit dieser Kenntnis kann PLACEPROPOSALS untersucht werden. Offensichtlich kann die while-Schleife ab Zeile 3 in $\mathcal{O}(nm)$ implementiert werden. Das Innere der for-Schleife ab Zeile 6 wird von der Ausführung von CHECKPROPOSALS und der for-Schleife in den Zeilen 8 bis 12 dominiert. Diese beiden Stellen benötigen jeweils eine Laufzeit von $\mathcal{O}(n)$. Die Konstruktionen in den Zeilen 13 und 15 benötigen demgegenüber nur $\mathcal{O}(\log n)$. Somit läuft die geschachtelte for-Schleife in den Zeilen 6 bis 23 in einer Laufzeit von $\mathcal{O}(nm)$ ab. Die äußere while-Schleife und somit die gesamte Prozedur läuft in $\mathcal{O}(n^2m)$ Zeit ab. Diese Ergebnis wird bestätigt, wenn man beachtet, dass der klassische „Men-propose-Women-dispose"-Algorithmus in $\mathcal{O}(n^2)$ Laufzeit implementiert werden kann und wir die Zahl der Kanten, die inzident zu einem Knoten aus P sind, von einer auf bis zu m Kanten ausgedehnt haben.

Das HUNGARIANUPDATE unterscheidet sich vom Ablauf her nicht von HUNGARIANUPDATE in [13]. Völlig analog zu dort kann unter Beachtung der Tatsache, dass wir nun $|E| = n \cdot m$ haben, ein Algorithmus zur Suche von Pfaden in $\mathcal{O}(nm)$ implementiert werden.

Wir können uns nun der Komplexitätsuntersuchung des Hauptalgorithmus annehmen. Die Initialisierungen zu Beginn des Algorithmus und die while-Schleife an dessen Ende können jeweils in $\mathcal{O}(nm)$ implementiert werden. Somit hängt offensichtlich die Komplexität des erweiterten HNS-Algorithmus vom Verhalten während der geschachtelten while-Schleifen ab.

Es gibt vier Ursachen, die zum Aufruf der inneren while-Schleife führen können. Es können fehlende Kapazitäten bei einem Arbeiter aufgefüllt werden, wobei die Obergrenze für die Anzahl solcher Fälle bei m liegt. Bestehende rigide Kanten eines Arbeiters können in höchstens m Fällen gelöscht werden. Rigide Kanten können nm-mal hinzugefügt werden. Kapazitäten können jeweils von einer von n Firmen als insolvent markiert werden. Die Laufzeitbeschränkung zum Aufruf dieser while-Schleife ergibt sich somit aus nm. Innerhalb der Schleife dominiert SCALINGMAXFLOW mit einer Laufzeit von $\mathcal{O}(n^2m^2 \log k)$ (mit [3]). Somit läuft diese while-Schleife in $\mathcal{O}(n^3m^3 \log k)$ ab.

Wenn nun durch HUNGARIANUPDATE neue Kanten in den Digraphen $G^{\sigma,v}$ aufgenommen werden, können neue Netzwerke entstehen. Mit CAPACITYSCALING können wir mögliche Flüsse ermitteln. Die Anzahl dieser neu hinzukommenden Möglichkeiten ist wieder durch nm begrenzt. Somit ändert sich die Laufzeit des Algorithmus nicht und wir erhalten als Laufzeit des gesamten Algorithmus $\mathcal{O}(n^3m^3 \log k)$.

5.7 Verallgemeinerung des HNS-Algorithmus

In diesem Kapitel soll nun gezeigt werden, dass der b-HNS-Algorithmus auch tatsächlich eine Verallgemeinerung des HNS-Algorithmus ist.

Es ist offensichtlich, wie der Input des HNS-Algorithmus umgeformt werden muss, damit ein Input für den b-HNS-Algorithmus entsteht. Die Präferenzmatrizen A, B und C bleiben unverändert bestehen. Außerdem setzen wir

$$\begin{aligned} k_{ij}^F = k_{ij}^R &= 1 \ \forall i \in P, j \in Q, \\ p_i^* &= 1 \ \forall i \in P, \\ q_j^* &= 1 \ \forall j \in Q. \end{aligned}$$

Das Outcome des klassischen HNS-Algorithmus können wir aus dem Outcome des b-HNS-Algorithmus ableiten. Nachdem im nun betrachteten Fall $|P| = |Q| = n$ ist, jeder Knoten genau eine Kapazität in Kanten unterbringen möchte und jede Kante die Kapazität 1 besitzt, ist sichergestellt, dass nach Abschluss des b-HNS-Algorithmus ein Matching vorliegt. Angenommen, im Outcome sind $i' \in P$ und $p' \in Q$ ungematcht. Dann hat insbesondere die Kante $(i', j')^R$ noch eine freie Kapazität, kann also genau noch die freien Kapazitäten von i' und j' aufnehmen. Dann hätte aber der b-HNS-Algorithmus nicht beendet werden dürfen und wir erhalten einen Widerspruch.

Die Matching-Permutation $\sigma \in S_n$ wird aus den Matrizen F und R abgeleitet. Für ein beliebiges $i \in P$ ist von den Elementen $\sigma^R_{ij'}$, $\sigma^F_{ij'}$ für alle $j' \in Q$ genau eines 1, alle anderen Einträge sind 0. Nachdem aber offensichtlich auch für jedes beliebige $j \in Q$ genau ein Element von $\sigma^R_{ij'}$, $\sigma^F_{ij'}$ für alle $i' \in P$ den Wert 1 annimmt und alle anderen Werte gleich 0 sind, erhalten wir die Matching-Permutation σ recht einfach. Für alle $i \in P$ gilt:

$$\sigma(i) = j \Leftrightarrow \sigma^R_{ij} = 1 \text{ oder } \sigma^F_{ij} = 1$$

Ist die Matching-Permutation σ bekannt, lässt sich die Flexibilitätsabbildung f ermitteln. Es gilt nämlich:

$$f(i) = 1 \Leftrightarrow \sigma^F_{i\sigma(i)} = 1$$
$$f(i) = 0 \Leftrightarrow \sigma^R_{i\sigma(i)} = 1.$$

Wir müssen nun noch die Payoff-Funktionen u und v definieren. Dies kann ebenfalls recht einfach geschehen, nachdem jeder Agent genau zu einer Kante der Kapazität 1 inzident ist und wir σ und f kennen:

$$u_i := \begin{cases} u^R_{i\sigma(i)} & \text{falls } f(i) = 0 \\ u^F_{i\sigma(i)} & \text{falls } f(i) = 1 \end{cases} \tag{5.32}$$

$$v_{\sigma(i)} := \begin{cases} v^R_{i\sigma(i)} & \text{falls } f(i) = 0 \\ v^F_{i\sigma(i)} & \text{falls } f(i) = 1 \end{cases} \tag{5.33}$$

Nun müssen wir zeigen, dass die so aus dem Outcome des b-HNS-Algorithmus ermittelten Daten die Voraussetzungen an ein zulässiges und stabiles Outcome im klassischen HNS-Algorithmus erfüllen. Diese Voraussetzungen sind gemäß [13]:

$$\forall 1 \le i \le n: \ f(i) = 1 \ \Rightarrow u_i + v_{\sigma(i)} = c_{i\sigma(i)} \tag{5.34}$$
$$\forall 1 \le i \le n: \ f(i) = 0 \ \Rightarrow u_i = a_{i\sigma(i)} \text{ und } v_{\sigma(i)} = b_{i\sigma(i)} \tag{5.35}$$
$$\forall 1 \le i \le n \, \forall 1 \le j \le m: \ u_i + v_j \ge c_{ij} \tag{5.36}$$
$$\forall 1 \le i \le n \, \forall 1 \le j \le m: \ u_i \ge a_{ij} \text{ oder } v_j \ge b_{ij} \tag{5.37}$$

Sei ein Outcome $\left(F, R, U^F, U^R, V^F, V^R\right)$ des b-HNS-Algorithmus gegeben. Sei außerdem $1 \le i \le n$ mit $f(i) = 1$. Dann ist mit (5.32) und (5.33) $u_i = u^F_{i\sigma(i)}$ und $v_{\sigma(i)} = v^F_{i\sigma(i)}$. Aus (5.3) folgt wegen $j = \sigma(i)$ und $\sigma^F_{i\sigma(i)>0}$ die folgende Aussage.

$$u_i + v_{\sigma(i)} = u^F_{i\sigma(i)} + v^F_{i\sigma(i)} = c_{i\sigma(i)}$$

und damit gilt Aussage (5.34).

Gelte nun bei gleichen Voraussetzungen $f(i) = 0$. Mit (5.32) und (5.33) ist dann $u_i = u^R_{i\sigma(i)}$ und $v_{\sigma(i)} = v^R_{i\sigma(i)}$. Mit (5.4) gilt dann analog zur eben bewiesenen Aussage:

$$\begin{aligned} f(i) = 0 \Rightarrow \sigma^R_{i\sigma(i)} > 0 &\Rightarrow u^R_{i\sigma(i)} = a_{i\sigma(i)} \text{ und } v^R_{i\sigma(i)} = b_{i\sigma(i)} \\ &\Rightarrow u_{i\sigma(i)} = a_{i\sigma(i)} \text{ und } v_{i\sigma(i)} = b_{i\sigma(i)} \end{aligned}$$

und somit auch Aussage (5.35).

Seien nun $i \in P$ und $j \in Q$ beliebig. Wenn $j = \sigma(i)$ im Outcome gilt, dann folgen die Aussagen (5.36) und (5.37) aus (5.34) und (5.35). Nehmen wir daher an, i und j sind im Outcome nicht gematcht. Dann können wir (5.9) und (5.8) anwenden, da deren Voraussetzungen offensichtlich gegeben sind.

Denn da i zu genau einem Knoten gematcht ist und $\sigma(i) \neq j$ gilt, ist

$$\begin{aligned} u^R_{min,j}(i) = u^F_{min,j}(i) = u_i \text{ und} \\ v^R_{min,i}(j) = v^F_{min,i}(j) = v_j \end{aligned}$$

Und somit ergibt sich

$$c_{ij} \leq u^F_{\min,j}(i) + v^F_{\min,i}(j) = u_i + v_j$$

und damit Aussage (5.36). Mit (5.8) folgt darüber hinaus sofort (5.37) und damit haben wir gezeigt, dass der b-HNS-Algorithmus tatsächlich eine Erweiterung des klassischen HNS-Algorithmus ist.

5.8 Verallgemeinerung des Arbeitsmarktspiels

Wie bereits im Kapitel 4 angekündigt, soll an dieser Stelle bewiesen werden, dass mit dem b-HNS-Algorithmus ein paarweise stabiles Outcome für das Arbeitsmarktspiel von Sotomayor erzeugt werden kann.

Ein Arbeitsmarkt ist nach Sotomayor über das 5-Tupel $M = (F, W, c, a, b)$ definiert. Dabei bezeichnen F die Menge der Firmen und W die Menge der Arbeiter und entsprechen den Mengen P und Q im Modell für den b-HNS-Algorithmus. Die $m \times n$-Matrix c beschreibt die Produktivität alle möglichen Partnerschaften und a und b sind die zur Verfügung stehenden Arbeitszeiteinheiten der Firmen und der Arbeiter.

Um die Bezeichnungen aus Kapitel 5.2.1 zu verwenden, setze $A = B = 0$, $C = c$, $P_{cap} = a$ und $Q_{cap} = b$. Im Modell von Sotomayor gibt es ausschließlich flexible Kanten, die keinen Kapazitätsrestriktionen unterliegen. Aus dem Grund wird im b-HNS-Algorithmus $K^R = 0$ gesetzt. Sei μ das Maximum aller Elemente von P_{cap} und Q_{cap}. Setze $k^F_{ij} = \mu$ für alle $1 \leq i \leq n$ und $1 \leq j \leq m$. Damit sind die Beschränkungen bei den Kantenkapazitäten so hoch angesetzt, dass sie wie beim Arbeitsmarktspiel keinen Einfluss auf das Outcome haben.

Damit ist gezeigt, dass jedes Arbeitsmarktspiel in Input-Daten des b-HNS-Algorithmus überführt werden kann. Führt man den b-HNS-Algorithmus mit diesen Daten durch, kann man daraus wie folgt ein Outcome (x, u, v) des Arbeitsmarktspiels generieren.

Setze für die Arbeitsallokation $x = F$. Denn F beinhaltet genau die Informationen, welche Agenten mit welcher Anzahl an Arbeitszeiteinheiten miteinander kooperieren.

Die Geldaufteilung $(u, v) \in \mathbb{R}^m \times \mathbb{R}^n$ wird aus den Matrizen U^F und V^F ermittelt. Setze dazu einfach

$$u_i = \sum_{j'=1}^{n} u^F_{ij'} \quad \forall i \in F$$

$$v_j = \sum_{i'=1}^{m} v^F_{i'j} \quad \forall j \in W.$$

Damit haben wir gezeigt, dass der b-HNS-Algorithmus eine Verallgemeinerung des Arbeitsmarktspiels von Sotomayor ist.

5.9 Anwendung auf das Kardinalitätsmatching

Nehmen wir nun in Analogie zu [12] an, dass für die rigiden Präferenzmatrizen $A = B = 0$ gilt, und für alle Einträge der flexiblen Präferenzmatrix C gelte $c_{ij} \in \{0, 1\}$. In diesem Fall reduziert sich das Problem darauf, möglichst viele Kapazitäten der Agenten auf den Kanten mit Profit 1 unterzubringen.

Beim Kardinalitätsmatching-Problem geht es darum, in einem gegebenen (bipartiten) Graphen ein Matching mit möglichst vielen Kanten herzustellen. Im klassischen Fall darf dabei jeder Knoten zu höchstens einer Kante inzident sein. Dieses Problem wird nun auf bipartite Graphen mit Vielfachheiten an den Knoten übertragen. Man erhält sozusagen ein „Kardinalitäts-b-Matching".

Der Algorithmus vereinfacht sich durch die Einschränkungen der Inputdaten deutlich. Dies betrifft insbesondere die while-Schleifen des b-HNS-Algorithmus. Wir stellen den modifizierten b-HNS-Algorithmus vor:

b-HNS-Algorithmus für Kardinalitätsmatching

1: **while** es gibt eine Firma $i \in P$, die noch solvente, freie Kapazitäten besitzt **do**
2: PROPOSE(i)
3: **end while**
4: Konstruiere den Digraphen $G^{\sigma,v}$
5: **while** es gibt ein $j_0 \in Q$, das überschüssige Kapazitäten hat **do**
6: **while** es gibt ein Netzwerk $\mathcal{N}$, der die Quelle j_0 besitzt und als Senke eine insolvente Firma besitzt oder ein $i_0 \in P$ und i_0 ist in $G^{\sigma,v}$ Anfangsknoten eines rigiden Pfeils oder einen Knoten j_1 aus Q, der noch offene Kapazitäten besitzt oder bereits ein rigides Angebot **do**
7: CAPACITYSCALING($\mathcal{N}$)
8: ALTERNATE ($\mathcal{N}$)
9: Aktualisiere $G^{\sigma,v}$ und K^F
10: **end while**
11: **end while**
12: **for all** $j \in Q$ **do**
13: **if** $\sum_{i \in P} k^F_{ij} > q^*_j$ then
14: Lösche willkürlich Kapazitäten aus zu j adjazenten Kanten bis $\sum_{i \in P} k^F_{ij} = q^*_j$ gilt
15: **end if**

16: **end for**
17: **for all** $j \in Q$ **do**
18: **for all** $i \in P$ **do**
19: $M \leftarrow M \cup \left(\{i,j\}^F, k_{ij}^F\right)$
20: **end for**
21: **end for**

Alle Tupel, die nach Ausführung des Algorithmus in der Menge M enthalten sind, beschreiben ein Matching maximaler Kardinalität im gegebenen bipartiten Graphen. Die Summe aller sich aus den Elementen von M ergebenden Kantenkapazitäten stellt die „b-Kardinalität“ des vorliegenden Graphen dar.

Literaturverzeichnis

[1] Ahuja, Ravindra K., Thomas L. Magnanti und James B. Orlin: *Network Flows: theory, algorithms and applications.* Prentice-Hall, Inc., 1993.

[2] Bayati, Mohsen, Devavrat Shah und Mayank Sharma: *Maximum Weight Matching via Max-Product Belief Propagation.* International Symposium on Information Theory 2005. Proceedings., Seiten 1763 – 1767, 2005.

[3] Bhattacharya, Binay: *CMPT 705: Chapter 7 - Network Flow.* http://www.cs.sfu.ca/ ssa121/personal/spring08/705/nflow.pdf, 2008. Gefunden am 07.05.2009.

[4] Dantzig, George Bernard: *Linear Programming and Extensions.* Princeton University Press, 1963.

[5] Eriksson, Kimmo und Johan Karlander: *Stable matching in a common generalization of the marriage and assignment models.* Discrete Mathematics, 217:Seiten 135 – 156, 2000.

[6] Fleiner, Tamás: *A matroid generalization of the stable matching polytope.* Proceedings of the 8th International IPCO Conference on Integer Programming and Combinatorial Optimization. LNCS 2081, Seiten 105 – 114, 2001.

[7] Fleiner, Tamás: *On the stable b-matching polytope.* Technischer Bericht TR-2002-03, Egervary Research Group on Combinatorial Optimization, 2002.

[8] Fujishige, Satoru und Akihisha Tamura: *A General Two-Sided Matching Market with Discrete Concave Utility Functions.* RIMS Preprint No. 1401, Kyoto University, 2003.

[9] Fujishige, Satoru und Akihisha Tamura: *A Two-Sided Discrete-Concave Market with Possibly Bounded Side Payments: An Approach by Discrete Convex Analysis.* Mathematics of Operations Research, 32(1):Seiten 136 – 155, 2007.

[10] Gale, David und Lloyd S. Shapley: *College Admissions and the stability of marriage.* American Mathematical Monthly, 69:Seiten 9 – 15, 1962.

[11] Galil, Zvi: *Efficient Algorithms for Finding Maximum Matching in Graphs.* ACM Computing Surveys, 18(1):Seiten 23 – 28, 1986.

[12] Hochstättler, Winfried, Hui Jin und Robert Nickel: *Note on an Auction Procedure for a Matching Game in Polynomial Time.* AAIM Proceedings, 2006.

[13] Hochstättler, Winfried, Robert Nickel und David Schiess: *Mixed Matching Markets.* Technischer Bericht feU-dmo010.08, FernUniversität in Hagen, 2008.

[14] Huang, Bert und Tony Jebara: *Loopy belief propagation for Bipartite Maximum b-Matching.* Artificial Intelligence and Statistics (AISTATS), 2007.

[15] Jin, Hui: *Algorithmen für Matching-Märkte.* Diplomarbeit, Lehrstuhl Mathematische Grundlagen der Informatik an der Brandenburgischen Technischen Universität Cottbus, 2005.

[16] Jungnickel, Dieter: *Graphs, Networks and Algorithms.* Springer Verlag, 2008.

[17] Kschischang, Frank R., Brendan J. Frey und Hans-Andrea Loeliger: *Factor Graphs and the Sum-Product Algorithm.* IEEE Transactions on Information Theory, 47(2), 2001.

[18] Kuhn, Harold W.: *The Hungarian method for the assignment problem.* Naval Research Logistics Quarterly, 2:Seiten 83 – 97, 1955.

[19] Kusserow, Martin und Ralph Hänsel: *Formelsammlung zur Diskreten Mathematik: 4.Sem. PD Dr. Labahn.* http://wwwstud.informatik.uni-rostock.de/inti/lehre/dma/fs04.pdf, 2003. Gefunden am 02.05.2009.

[20] Reiss, Kristina und Gerald Schmieder: *Basiswissen Zahlentheorie: Eine Einführung in Zahlen und Zahlbereiche.* Springer Verlag, Berlin, 2. Auflage, 2007.

[21] Roth, Alvin E.: *Stability and polarization of interest in job matching.* Econometrica, 53:Seiten 47 – 57, 1984.

[22] Roth, Alvin E. und Marilda Sotomayor: *The college admissions problem revisited.* Econometrica, 57(3):Seiten 559 – 570, 1989.

[23] Roth, Alvin E. und Marilda Sotomayor: *Two-Sided Matching: A Study in Game-Theoretic Modeling and Analysis.* Cambridge University Press, 1992.

[24] Rothblum, Uriel G.: *Characterization of stable matchings as extreme points of a polytope.* Mathematical Programming: Series A, 54:Seiten 57 – 67, 1992.

[25] Seip, Ulrich: *Manuskript zum Kurs Graphentheorie an der FernUniversität in Hagen*, Oktober 2007.

[26] Shapley, Lloyd S. und Martin Shubik: *The assignment game I: The core.* International Journal of Game Theory, 1:Seiten 111 – 130, 1972.

[27] Sotomayor, Marilda: *Three remarks on the many-to-many-stable matching problem.* Mathematical Social Sciences, 38(1):Seiten 55 – 70, 1999.

[28] Sotomayor, Marilda: *Existence of stable outcomes and the lattice property for a unified matching market.* Mathematical Social Sciences, 39:Seiten 119 – 132, 2000.

[29] Sotomayor, Marilda: *A labor market with heterogeneous firms and workers.* International Journal of Game theory, 31:Seiten 269 – 283, 2002.

[30] Tittmann, Peter: *Graphentheorie: Eine anwendungsorientierte Einführung.* Fachbuchverlag Leipzig im Carl Hanser Verlag, 2003.

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Großheirath, 26.Mai 2009 Markus Werner